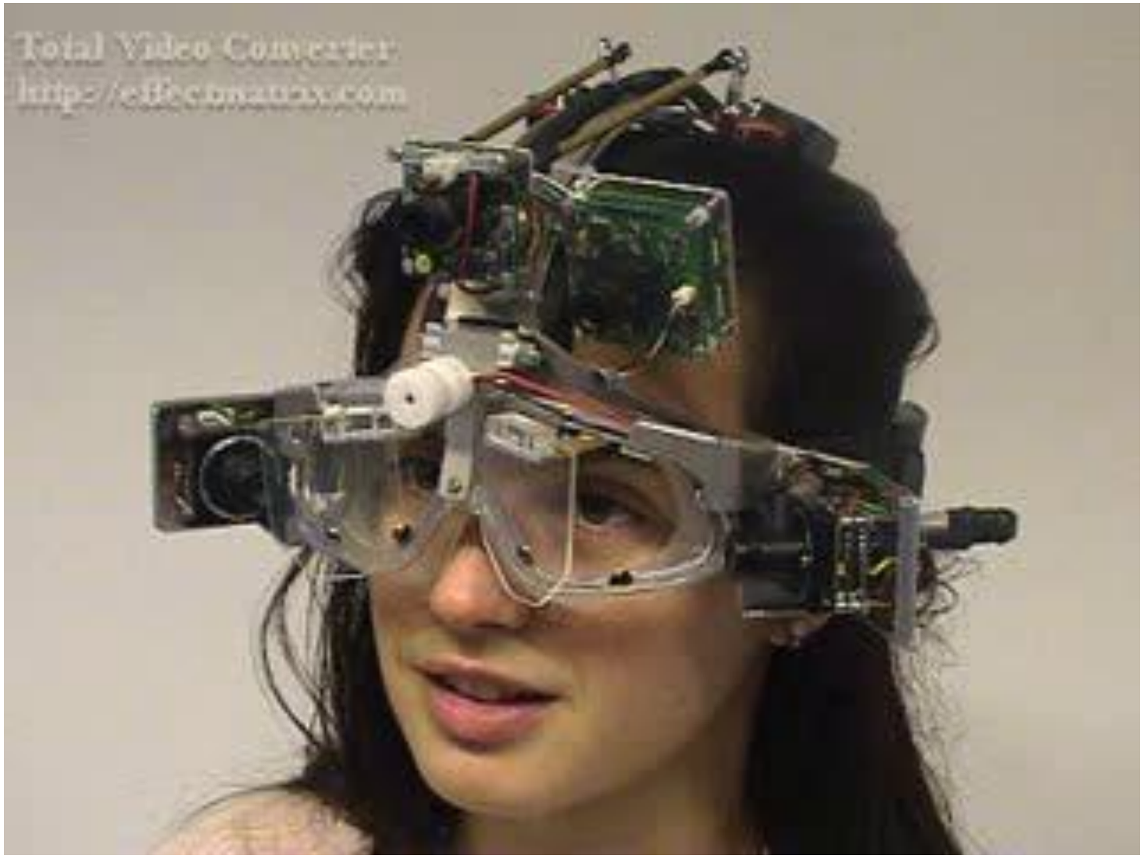
Image Processing for Robotics

Pixel: Pool Playing Robot

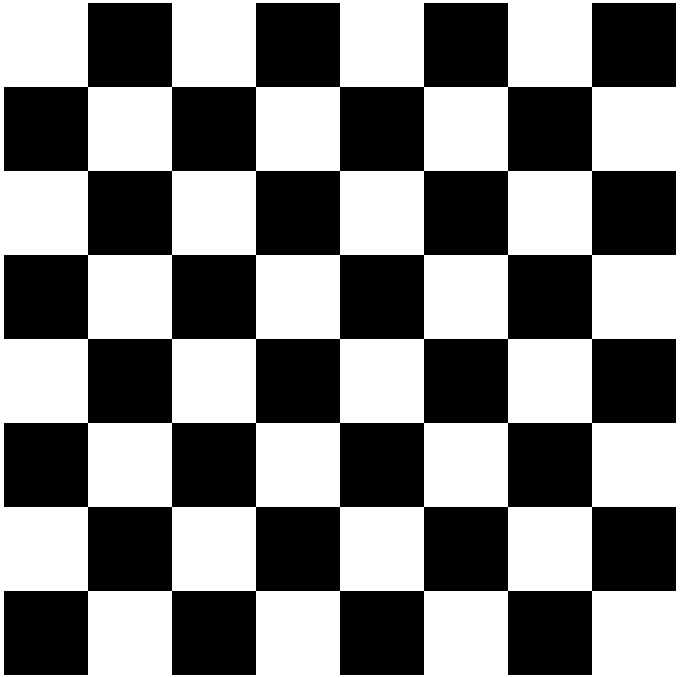
Eye-Gaze Tracking



Image?

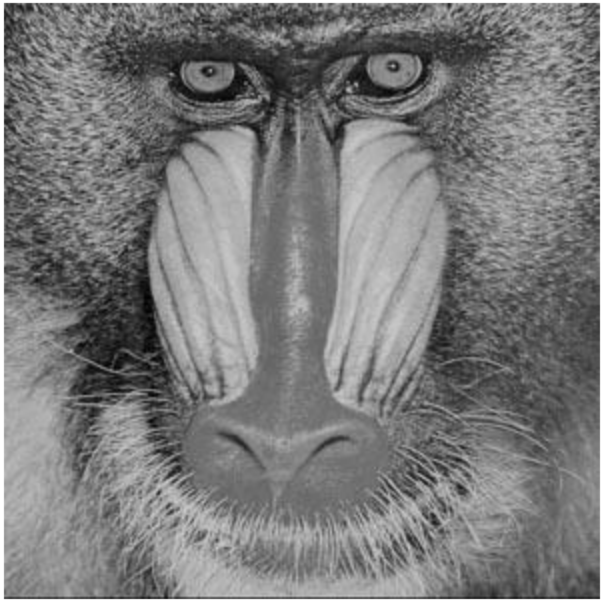
- An array (or matrix) of square pixels arranged in columns and rows.
- Playing with matrices and numbers

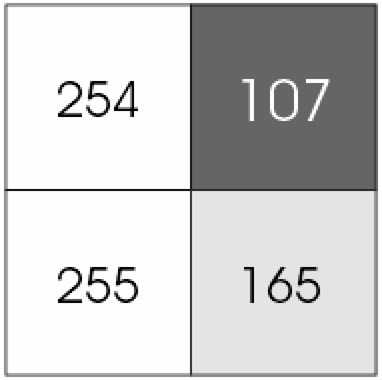
Binary Image



A **binary image** is a digital image that has only two possible values for each pixel. Typically the two colors used for a binary image are black and white though any two colors can be used

Grayscale Image

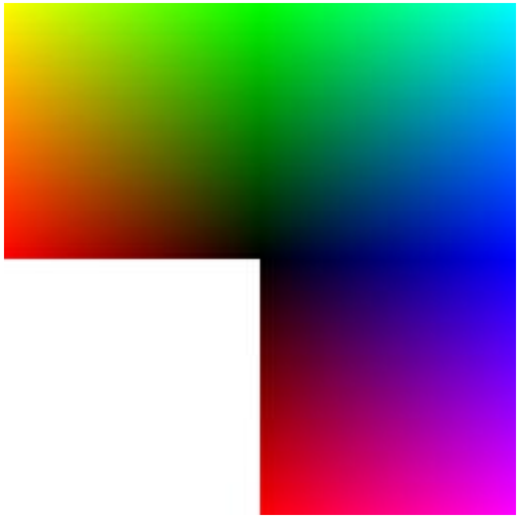




254	107
255	165

- An image with 8-bit color depth, with grayscale intensities varying from 0 to 255.

RGB Image



RGB: 24 bit per pixel, 3 eight bit unsigned integers (0 to 255) represent the intensities of red, green and blue.

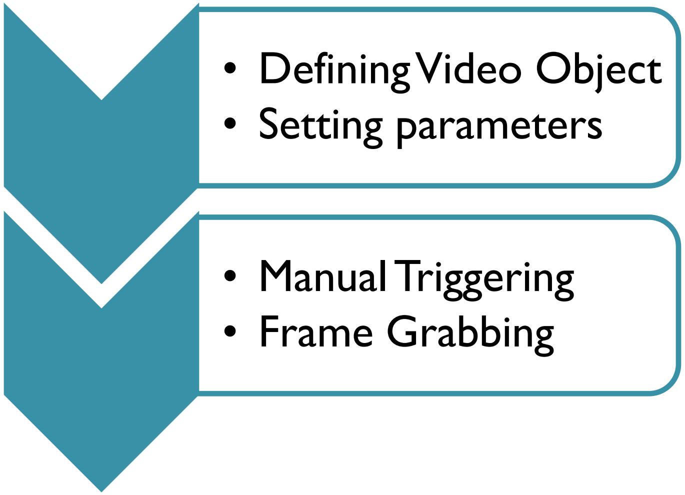
(0, 0, 0) is black
(255, 255, 255) is white
(255, 0, 0) is red
(0, 255, 0) is green
(0, 0, 255) is blue
(255, 255, 0) is yellow
(0, 255, 255) is cyan
(255, 0, 255) is magenta

Softwares for image processing

- Matlab Image Processing Toolbox
- Intel OpenCV Library
- CIMG Library

Image Processing with Matlab

Image Acquisition



- Defining Video Object
- Setting parameters

- Manual Triggering
- Frame Grabbing

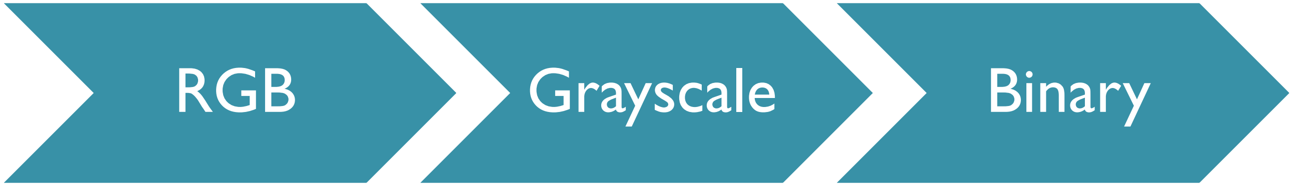
- Do not use the first few frames grabbed, the camera lamp needs to get warmed up.
- Remember always to flush the used frames logged on to the memory.

```
vid=videoinput('winvideo',1,'RGB24_320x240');
set(vid,'FramesPerTrigger',1);
set(vid,'TriggerRepeat',Inf);
triggerconfig(vid,'manual');

 trigger(vid);
 imrgb=getdata(vid,1);
```

```
stop(vid);
delete(vid);
clear vid;
```

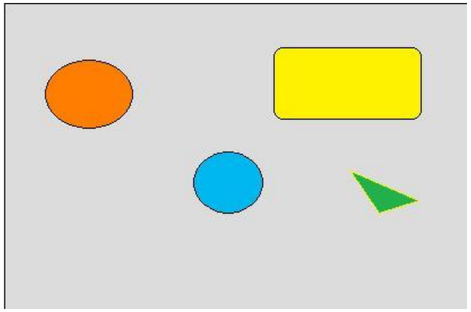
Color Space Conversion

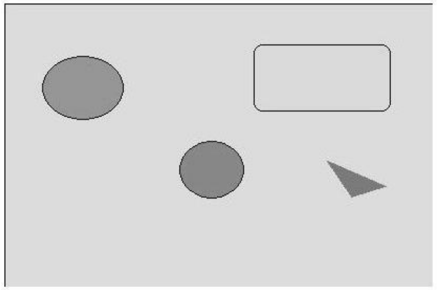


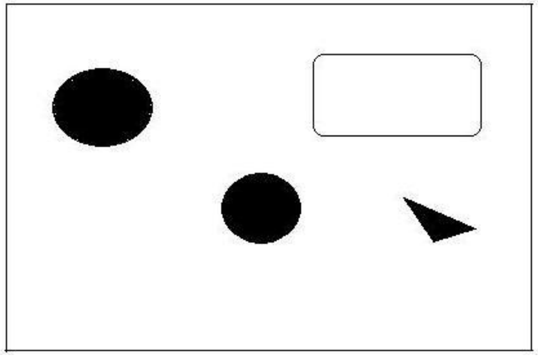
RGB

Grayscale

Binary

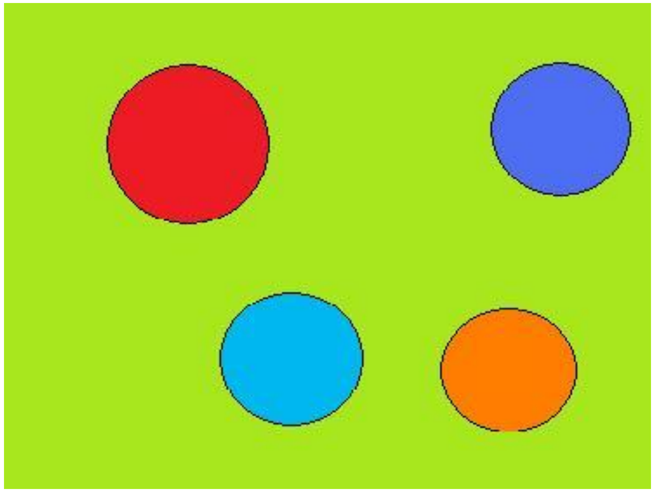


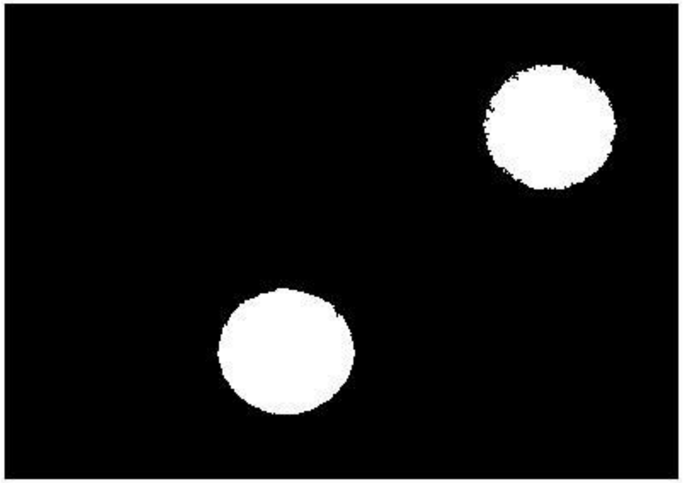




```
I = rgb2gray(imrgb);
threshold = graythresh(I);
bw = im2bw(I,threshold);
imshow(bw)
```

Image Thresholding Using RGB Image



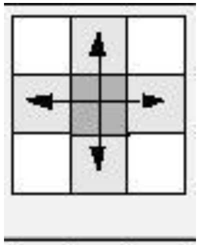


```
for i=1:279
        for j=1:396
            if((RGB(i,j,1)<100 ) && (RGB(i,j,2)>100 && RGB(i,j,2)<200) && (RGB(i,j,3)>200 ))
                GREY(i,j)=255;
            else
                GREY(i,j)=0;
            end
        end
    end
```

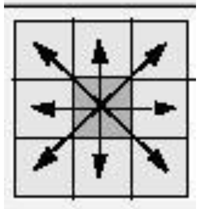
Labeling Connected Objects

```
[L,num] = bwlabel(BW,n)

BW =
   [1   1   1   0   0   0   0   0
    1   1   1   0   1   1   0   0
    1   1   1   0   1   1   0   0
    1   1   1   0   0   0   1   0
    1   1   1   0   0   0   1   0
    1   1   1   0   0   0   1   0
    1   1   1   0   0   1   1   0
    1   1   1   0   0   0   0   0];
```



4 Connected



8 Connected

```
L = bwlabel(BW,4)
L =

    1   1   1   0   0   0   0   0
    1   1   1   0   2   2   0   0
    1   1   1   0   2   2   0   0
    1   1   1   0   0   0   3   0
    1   1   1   0   0   0   3   0
    1   1   1   0   0   0   3   0
    1   1   1   0   0   3   3   0
    1   1   1   0   0   0   0   0
```

Evaluating Geometrical Properties

STATS = regionprops(L, properties)

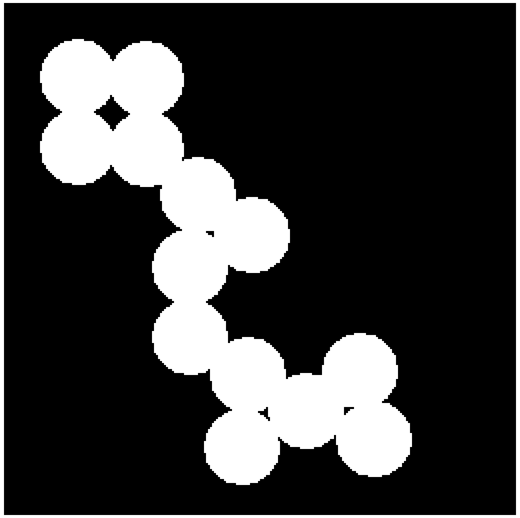
'Area'‘
EulerNumber'‘
Orientation'
'BoundingBox'‘
Extent'‘
Perimeter'
'Centroid'‘
Extrema'‘
PixelIdxList''
'ConvexArea'‘
FilledArea'‘
PixelList'
'ConvexHull'‘
FilledImage'‘
Solidity'

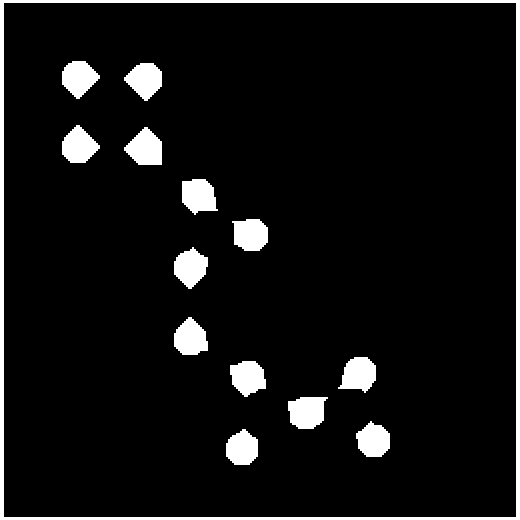
Example: Selecting Regions Based on Certain Criteria

```
idx = find([stats.Area] > 80);
BW2 = ismember(L,idx);
```

Morphological Operations

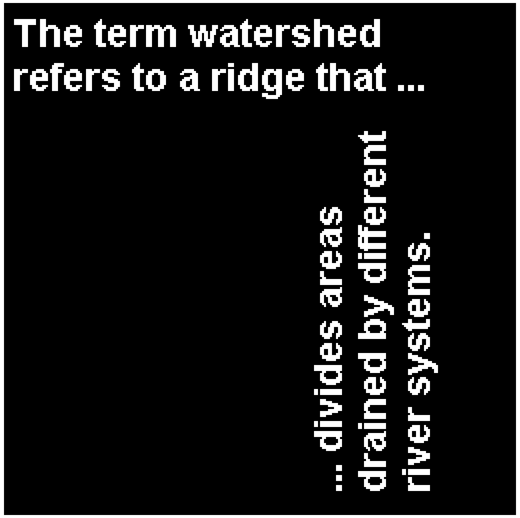
Eroding an Image

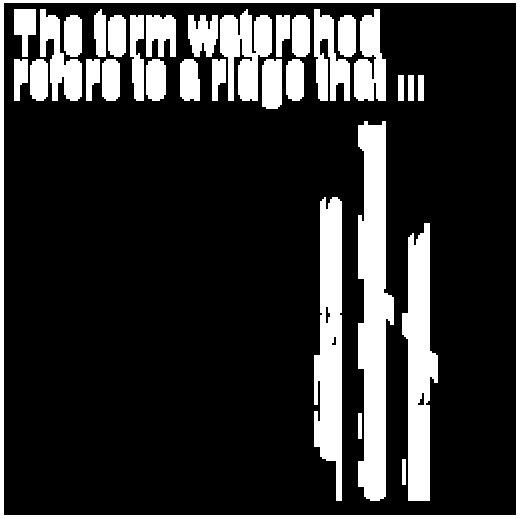




Morphological Operations

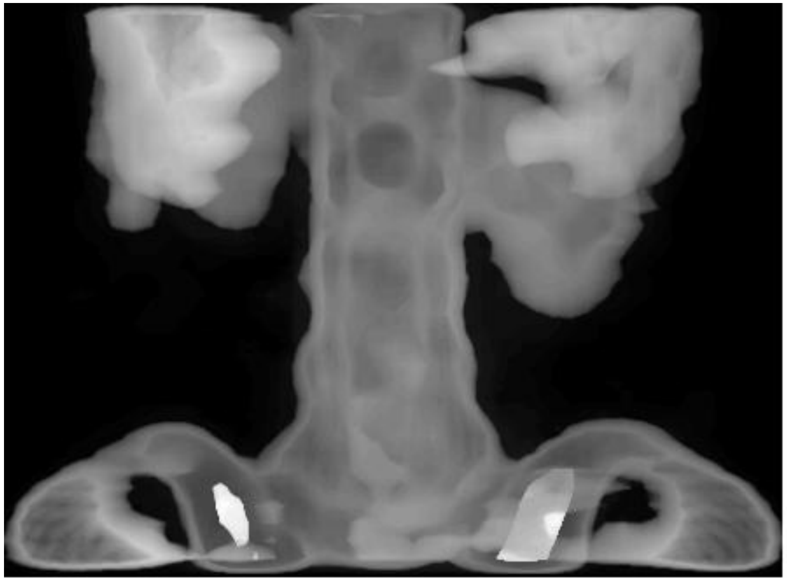
Dilating an Image



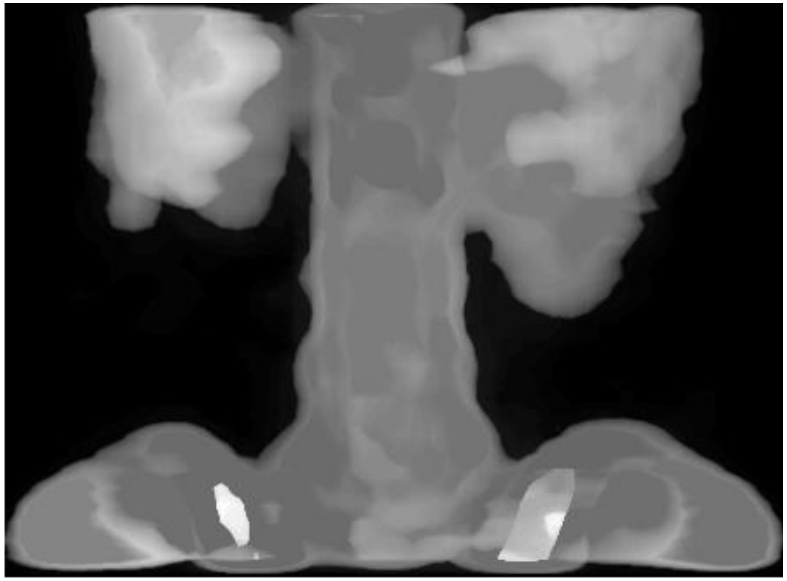


Morphological Operations

Filling Holes (*imfill*)

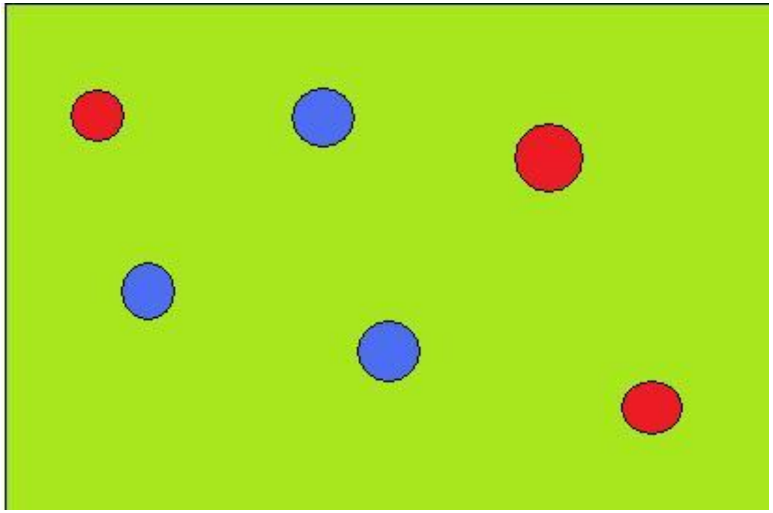


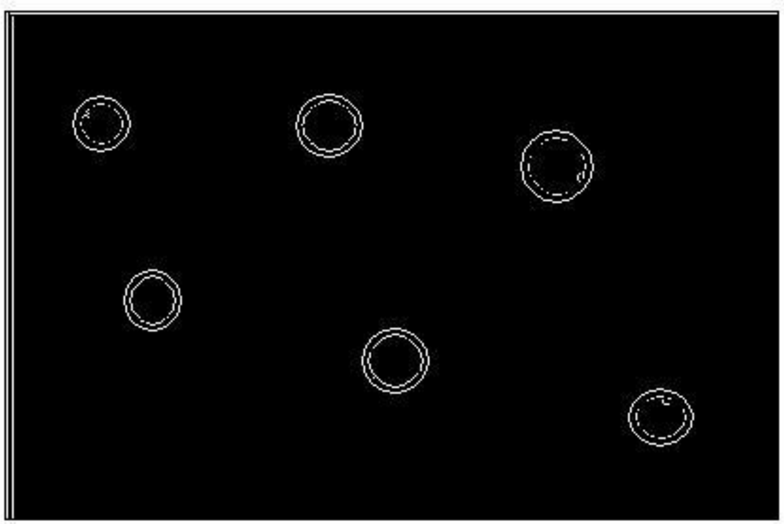
Original



After Filling Holes

Edge Detection

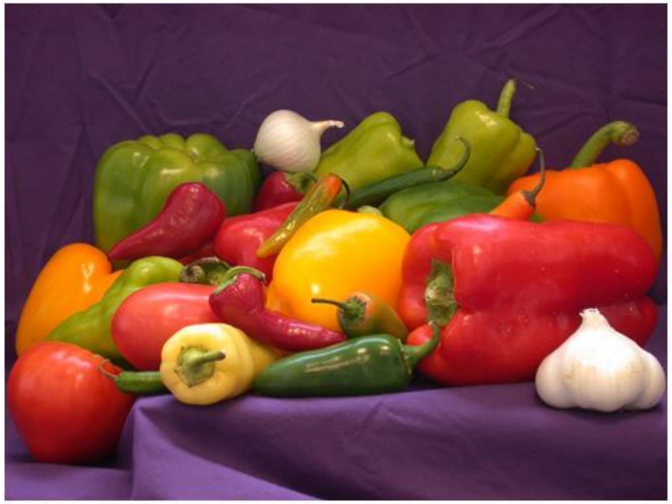


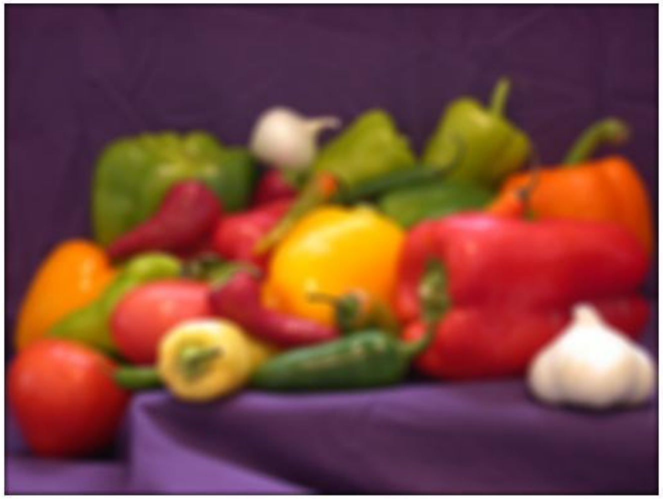


Canny, Sobel
Edge Operator

Basic idea: Finding the gradient of the image in the x and y directions. Around the boundary of the objects, the intensity varies steeply, hence a derivative of intensities results in the edge of the boundaries.

Image Enhancement



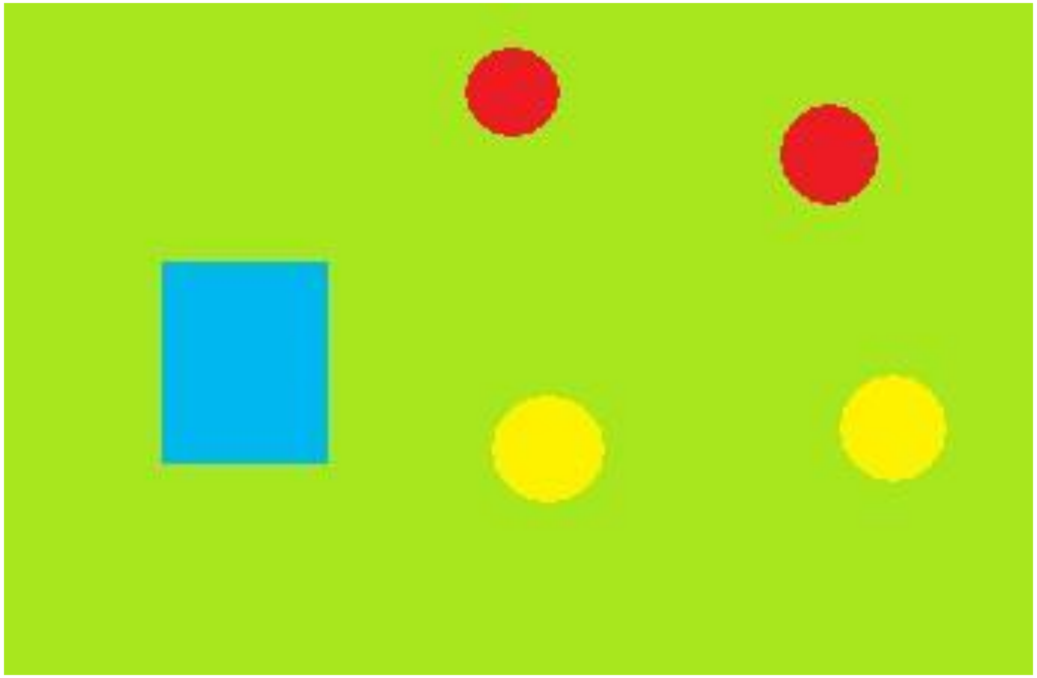


```
originalRGB = imread('peppers.png');
h = fspecial('disk', 5);
filteredRGB = imfilter(originalRGB, h);
imshow(filteredRGB)
```

Important Tools

- Cropping an image (*imcrop)*
- Transformations, rotation (*imrotate*)
- Image Profiling (*improfile*)
- Image Histogram (*imhist*)
- *imview*

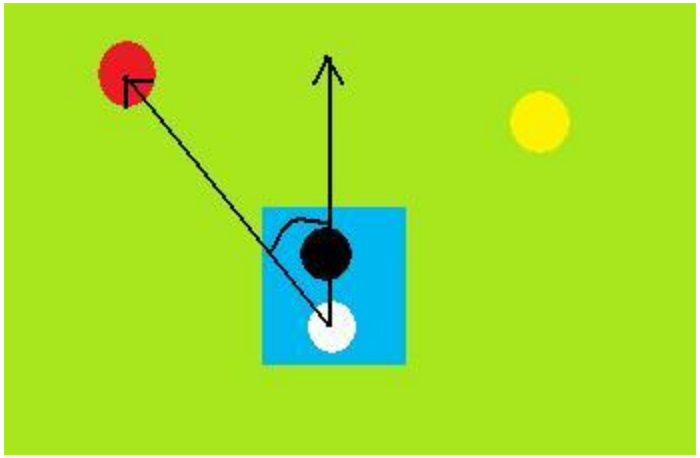
Identifying your Bot



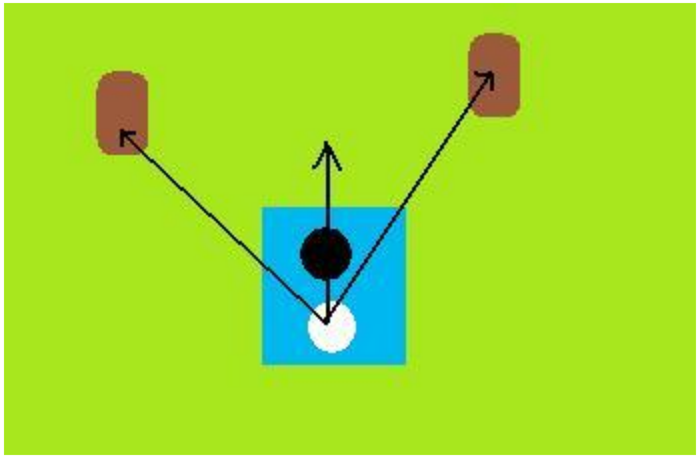
index= 4*pi*area/perimeter^2
Rectangle: 0.7 approx
Circles: >0.9

Orientation of the Bot

Ball Following Robot



Obstacle Avoiding Robot

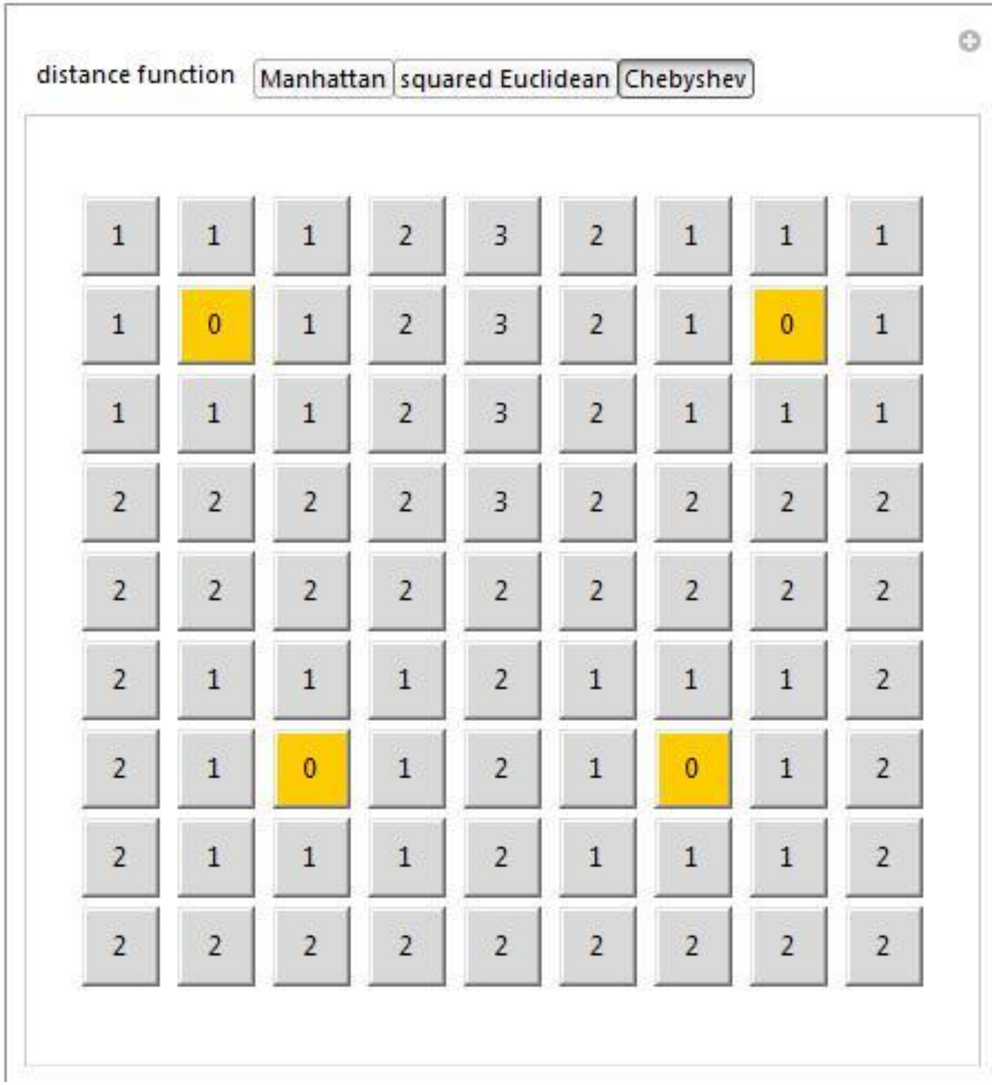


Angle of orientation: *atan/atan2*

Distance Transforms

- The distance transform gives the distance-map representation of an image. It replaces each pixel with the Euclidean distance between the pixel and the nearest background pixel
- Think of minesweeper!

DistanceTransforms



distance function Manhattan squared Euclidean [✓ Chebyshev]

1	1	1	2	3	2	1	1	1
1	0	1	2	3	2	1	0	1
1	1	1	2	3	2	1	1	1
2	2	2	2	3	2	2	2	2
2	2	2	2	2	2	2	2	2
2	1	1	1	2	1	1	1	2
2	1	0	1	2	1	0	1	2
2	1	1	1	2	1	1	1	2
2	2	2	2	2	2	2	2	2

Play with images
Explore Image Processing