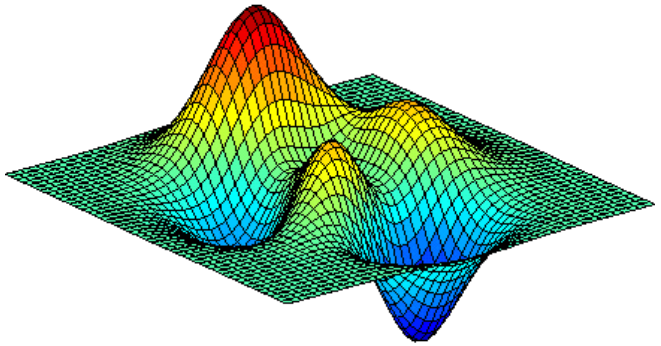


Introduction in

MATLAB

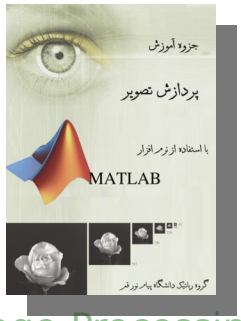
Image Processing Toolbox

By Shahriar Yazdipour

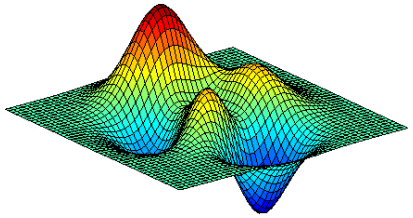
Agenda

- Applications of Image Processing
- Image Pre processing
- Image segmentation
- Colors
- IP Toolbox Functions
- Edge Detection
- Blur
- Demo
  - Display and working with tools
  - Colors
  - Blurring
  - Edge detection
  - Segmentation
  - Road line detection
  - Sudoku Solver

Resources and demo files



- **Demos + PPT + eBook**
- http://www.slideshare.net/ShahriarYazdipour/
- https://github.com/yazdipour/Intro-Matlab-Image-Processing
- http://shahriar.in/blog/post/141



DOCUMENTS

http://mathworks.com/matlab

http://mathworks.com/matlabcentral

https://www.mathworks.com/products/image.html

Applications of Image Processing

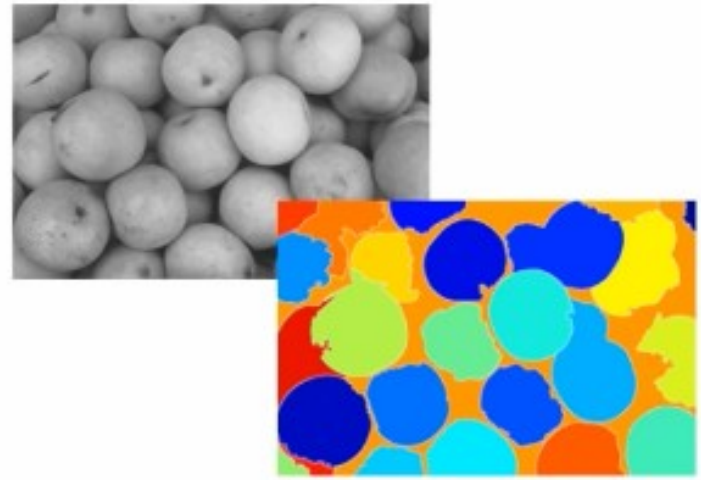
- Robotics
- Medical imaging
- Automotive safety
- Geospatial computin
- Surveillance
- And more …

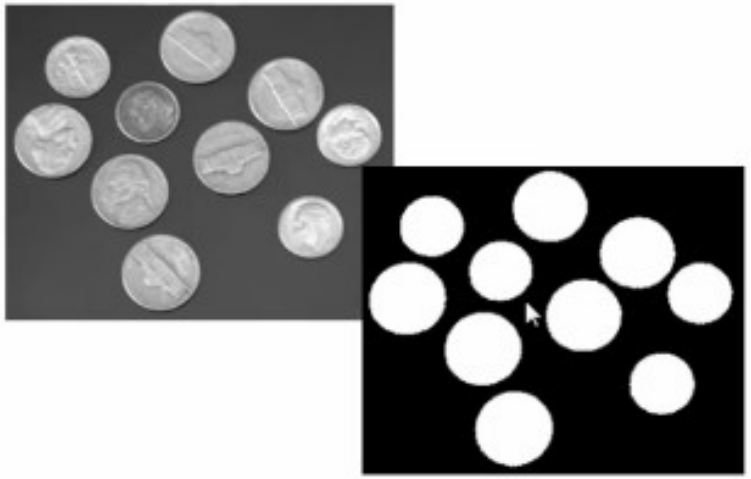
Pre Processing

- Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further processing.
- Such as:
  - Reduce noise
  - Deblurring
  - Brightness Adjusting
  - Color Adjusting
  - ...

Image segmentation

- Image segmentation is the process of dividing an image into multiple parts. That is typically used to identify objects and other relevant information within an image.

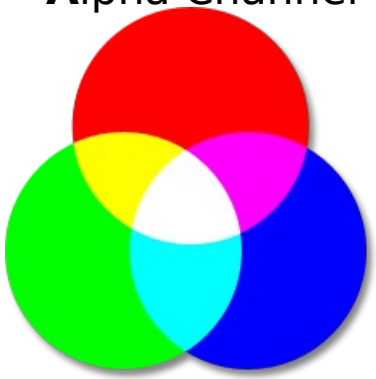




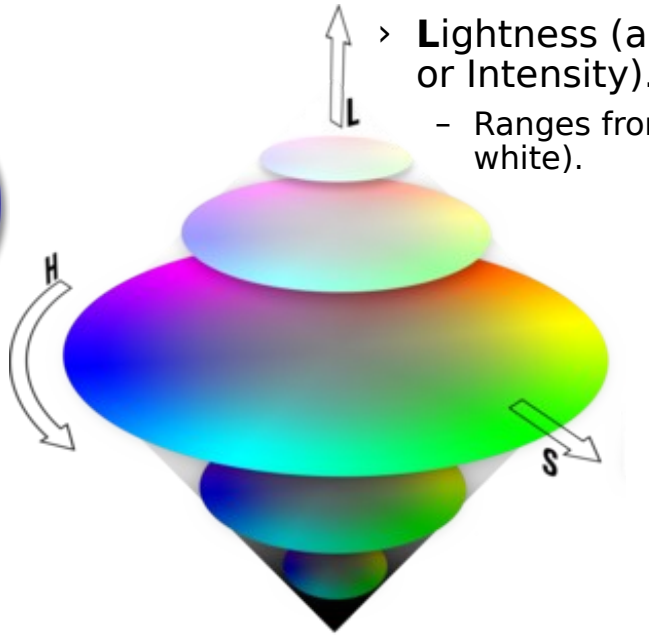
Colors

http://www.colorizer.org/

- **RGB (A)**
- **R**ed, which ranges from 0-255
- **G**reen, which ranges from 0-255
- **B**lue, which ranges from 0-255
- **A**lpha Channel for transparency

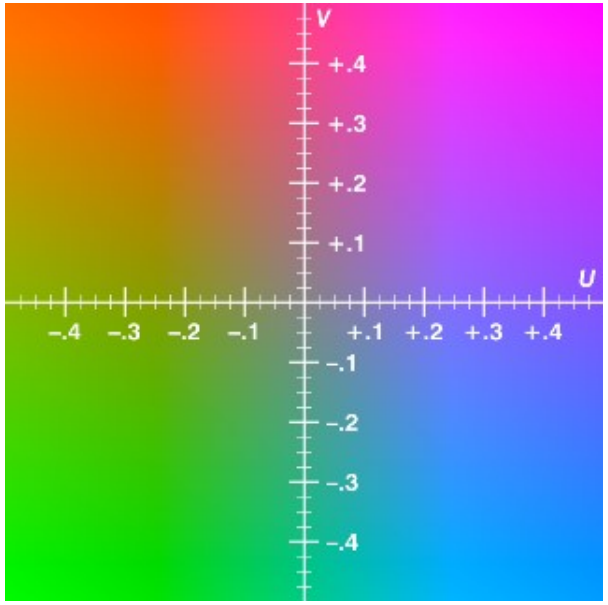


- **HSL**
- **H**ue : the color type (such as red, blue,..)
  Ranges from 0 to 360° in most applications (each value corresponds to one color : 0 is red, 45 is a shade of orange and 55 is a shade of yellow).
- **S**aturation : variation of the color depending on the lightness.
  - Ranges from 0 to 100% (center = lightest).
- **L**ightness (also Luminance or Luminosity or Intensity).
  - Ranges from 0 to 100% (from black to white).



Colors

- **YUV**
- **Y**, the luma component, or the brightness.
  - Ranges from 0 to 100% in most applications.
- **U** and **V** are the chrominance components



- **CMYK**
- **C**yan.
  - Ranges from 0 to 100% in most applications.
- **M**agenta.
- **Y**ellow.
- blac**K**

IP Toolbox Functions

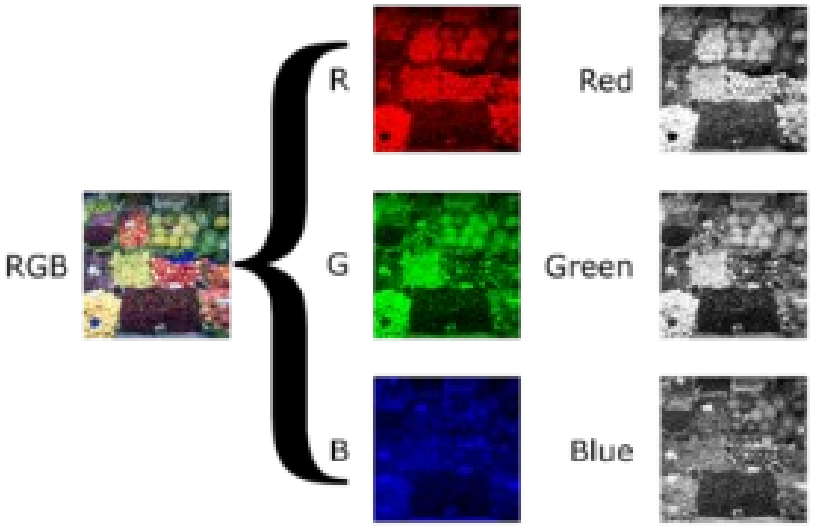
- `Imread(fileDir,[fileType])`
- `Imwrite(Matrix,fileDirToSave,[fileType])`
- `Imshow(M)`
- `Imtool(M)`

IP Toolbox Functions

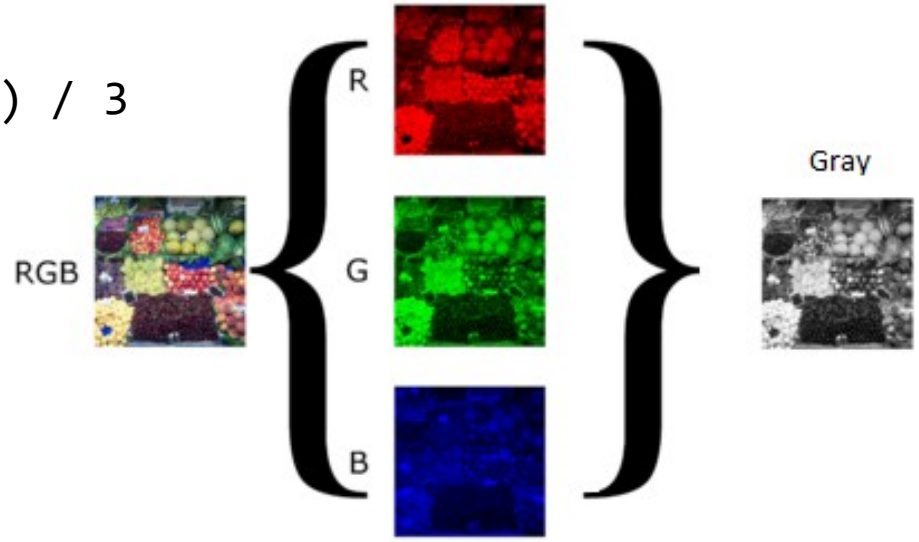
- Imcrop(I,[x,y,width,height]) %%submatrix
- Rgb2gray(I)
- Im2bw(I,level)
- Bwlabel(bw)
- Imfeature(I,'all') OR regionprops(I,'all')
  - Area / Centeroid / BoundingBox / Eccentricity
  - Orientation / PixelList /...

RGB to Gray

```
Gray = Red   = I[:,:,1]
Gray = Green = I[:,:,2]
Gray = Blue  = I[:,:,3]
```

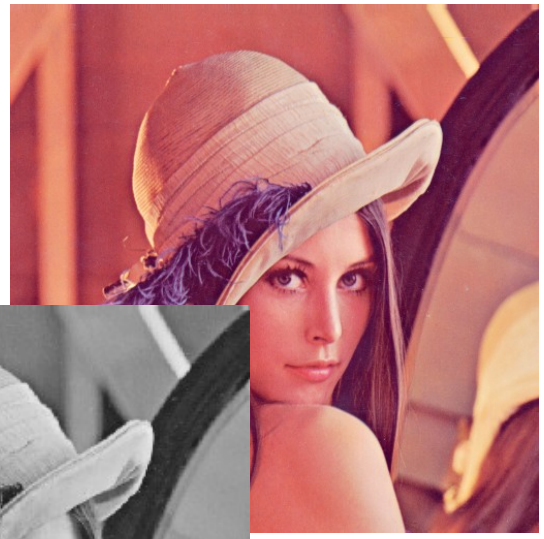


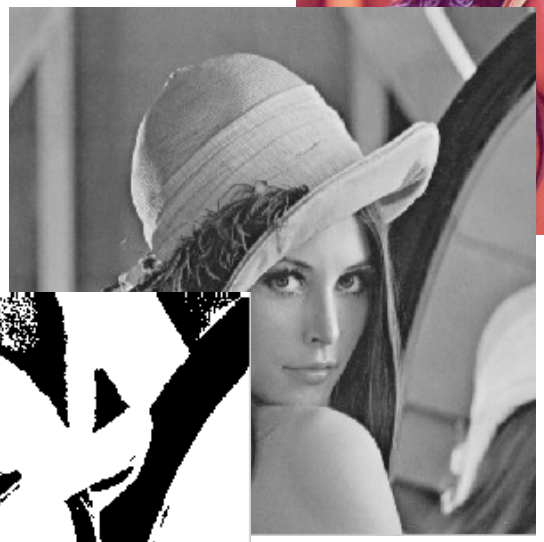

```
Gray = (Red + Green + Blue) / 3
```



RGB to Black & White

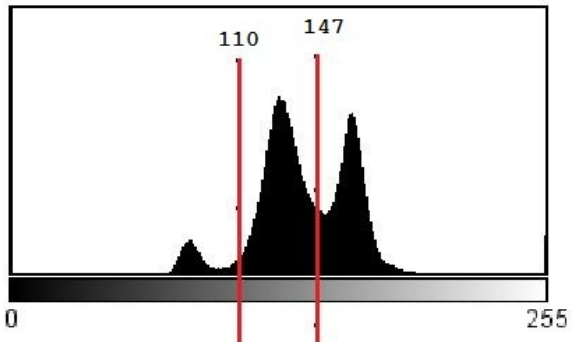
```
Bw= im2bw(Image,level)

Image must be a 2D Matrix.

Default Level: 0.5
```





Computing Level

```
if (level*255 > PixelVal)
PixelVal=1;
else PixelVal=0;
```



To compute the level argument,
we can use the function graythresh
The graythresh function uses Otsu's method, which chooses the threshold to minimize the intraclass variance of the black and white pixels.

Otsu
, N., "A Threshold Selection Method from Gray-Level Histograms," IEEE Transact ions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66
.

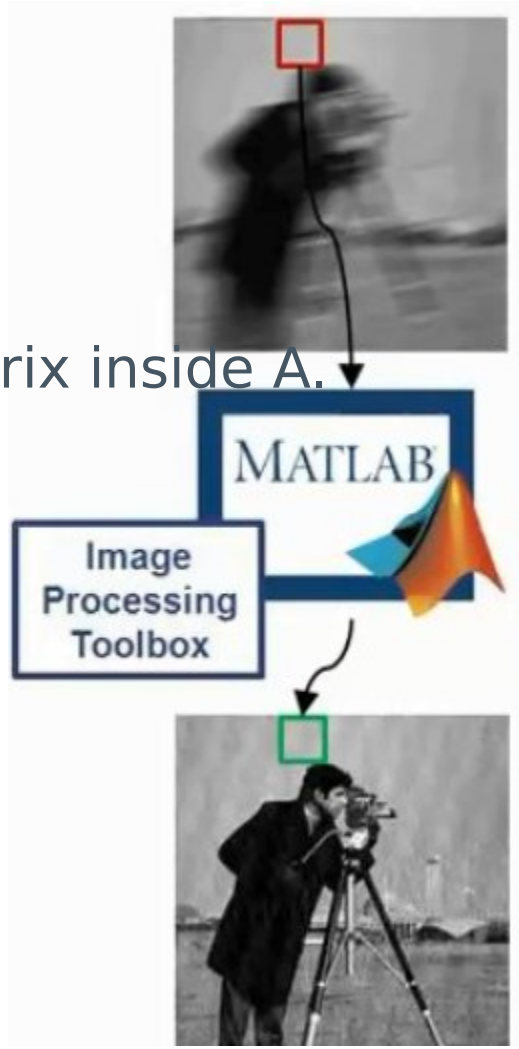
IP Toolbox Functions

```
B = blockproc(A,[M N],func)
```

Execute func foreach [M N] size matrix inside A.

Use it to:

- Manage speed and memory
- Sometimes more accurate



IP Toolbox Functions

- IPPL: Intel Integrated Performance Primitives Library
- Imadd
- Imdivide
- Immultiply
- Imabsdiff
- Imcomplement
- …

IP Toolbox Functions

Imnoise

`imnoise(I,'salt & pepper',d)` adds salt and pepper noise to the image I, where d is the noise density. This affects approximately d*numel(I) pixels. The default for d is 0.05. (On off method)

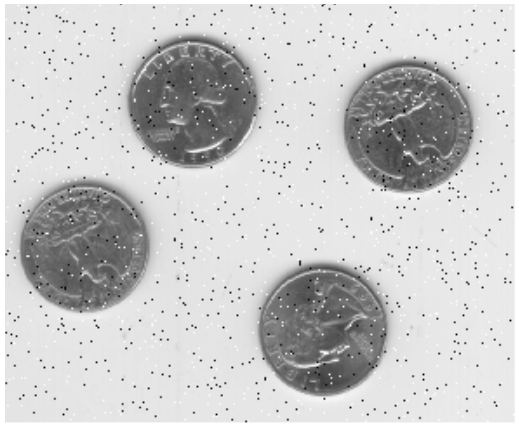
`imnoise(I,'speckle',v)` adds multiplicative noise to the image I, using the equation J = I+n*I, where n is uniformly distributed random noise with mean 0 and variance v. The default for v is 0.04.

`medfilt2(A,m,n)` :2-D median filtering

median filtering is a nonlinear operation often used in image processing to reduce "salt and pepper" noise. A median filter is more effective when the goal is to simultaneously reduce noise and preserve edges

Each output pixel contains the median value in the m-by-n neighborhood around the corresponding pixel in the input image. medfilt2 pads the image with 0s (Black) on the edges.

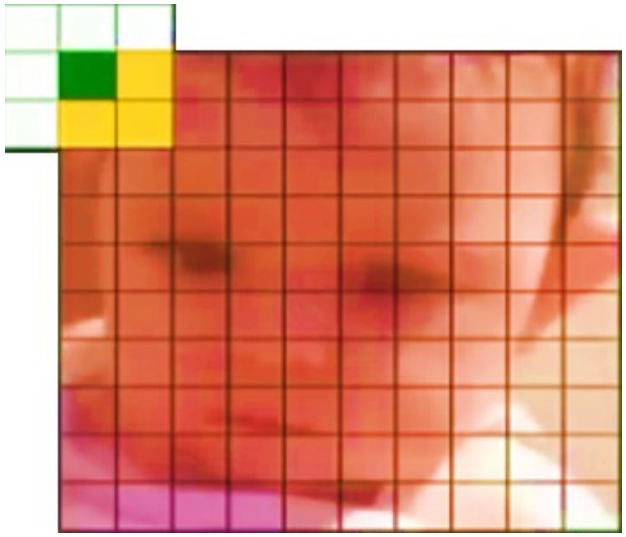
Salt & Pepper noise + medfild2



Blurring

Gaussian blur

```
PSF = fspecial('gaussian',3);
B=imfilter(I,PSF,'symmetric');
```



Gaussian blur

With weighed kernel

0.1	0.25	0.5	0.25	0.1
0.25	0.5	0.85	0.5	0.25
0.5	0.85	1	0.85	0.5
0.25	0.5	0.85	0.5	0.25
0.1	0.25	0.5	0.25	0.1

IP Toolbox Functions

`BW = edge(I,[thresh])` takes a grayscale or a binary image, and returns a binary image

The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of Image is maximum.

Roberts

Sobel

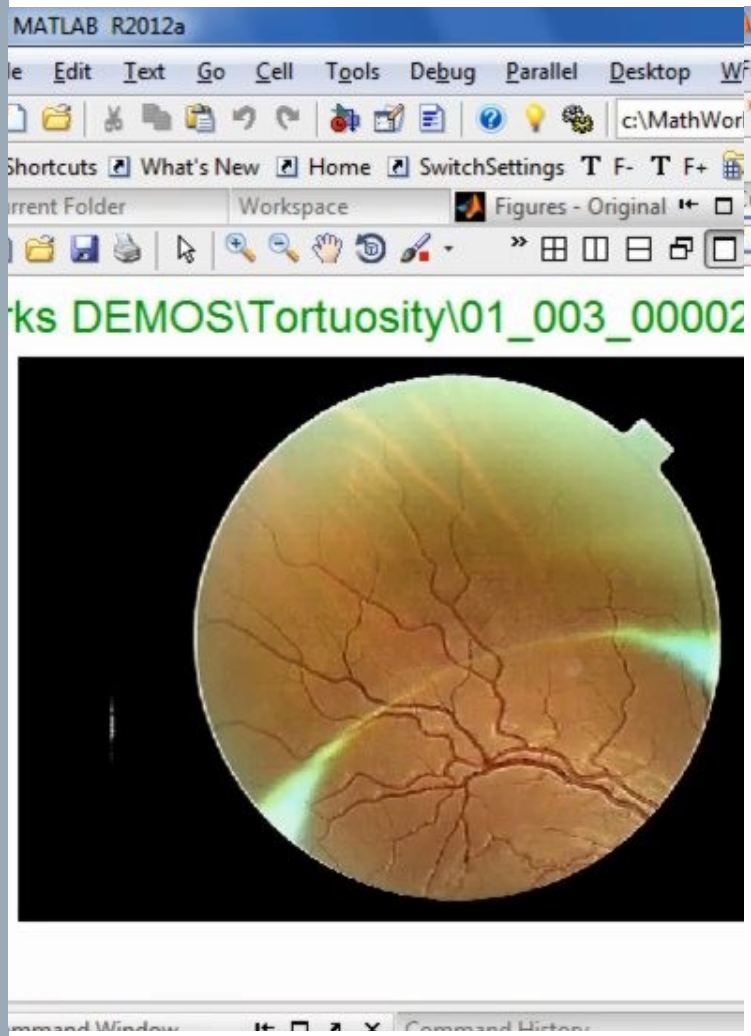
The Canny

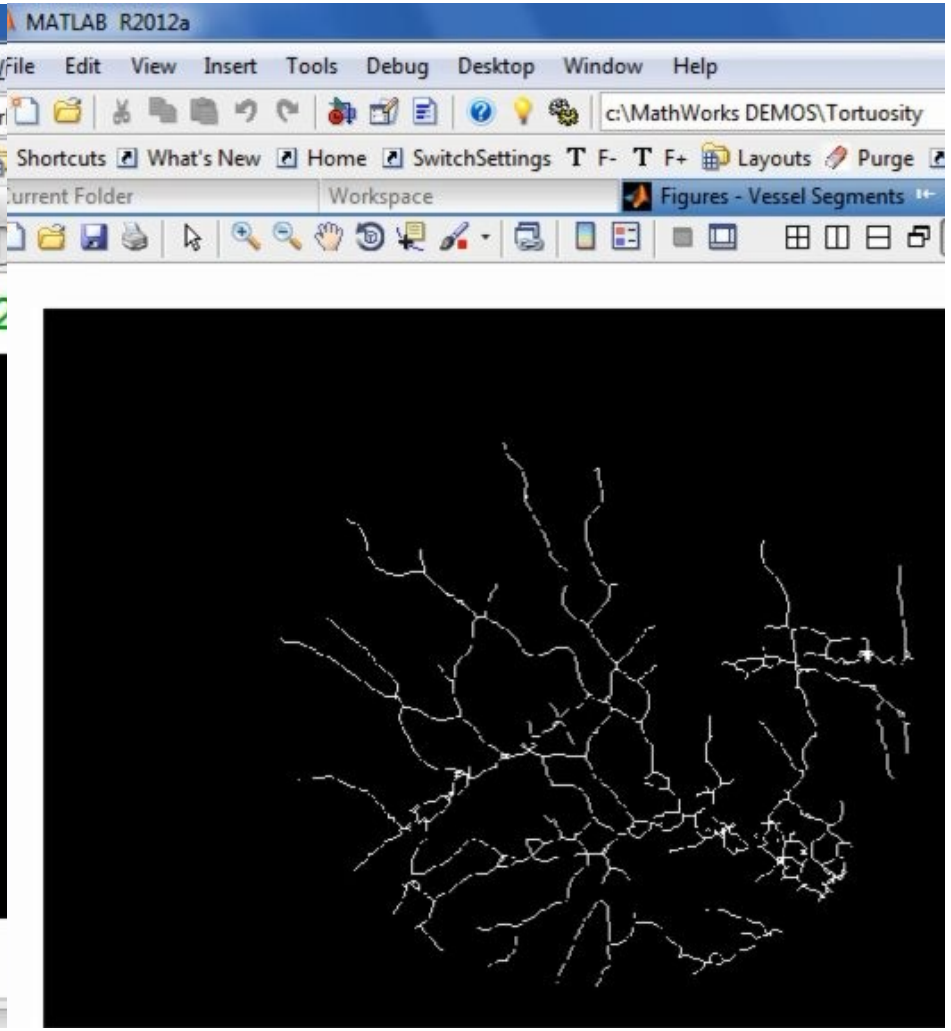
Canny, John, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol. PAMI-8, No. 6, 1986, pp. 679-698.

Lim, Jae S., Two-Dimensional Signal and Image Processing, Englewood Cliffs, NJ, Prentice Hall, 1990, pp. 478-488.

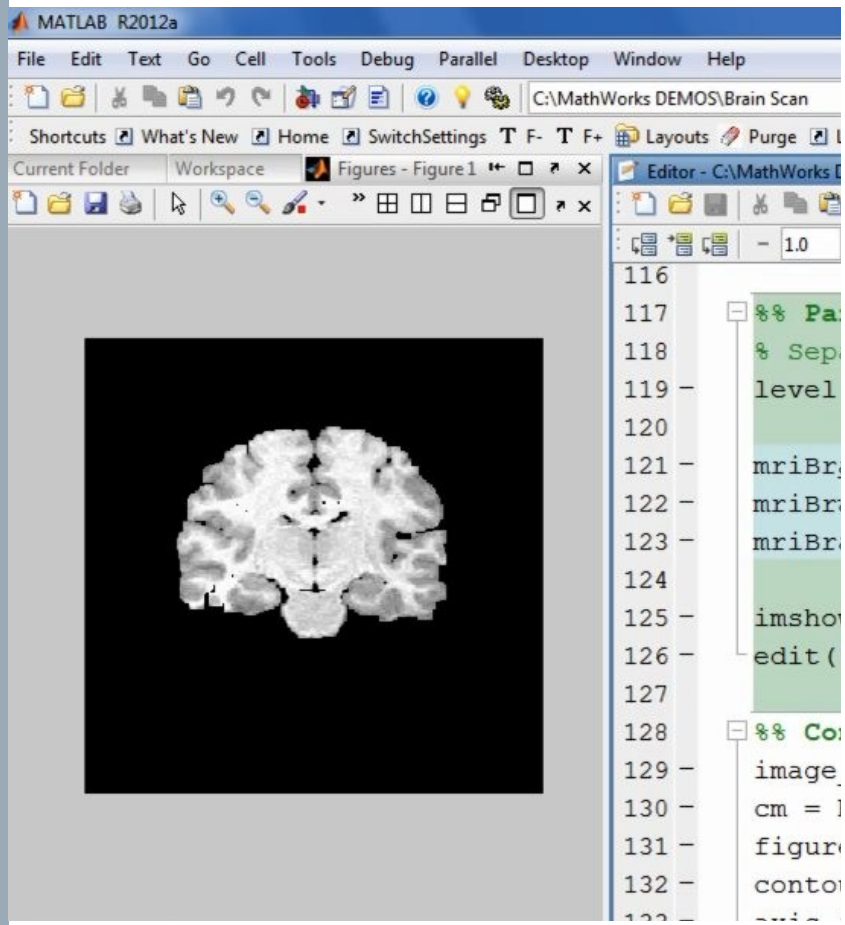
Parker, James R., Algorithms for Image Processing and Computer Vision, New York, John Wiley & Sons, Inc., 1997, pp. 23-29.

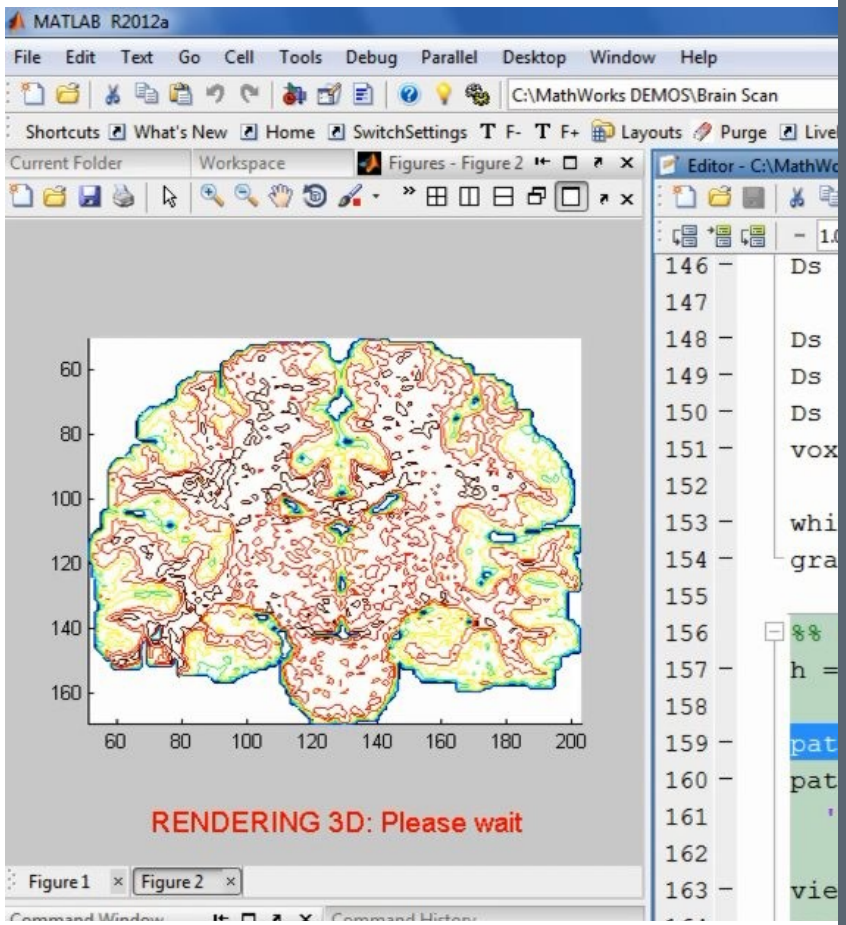
Edge Detection



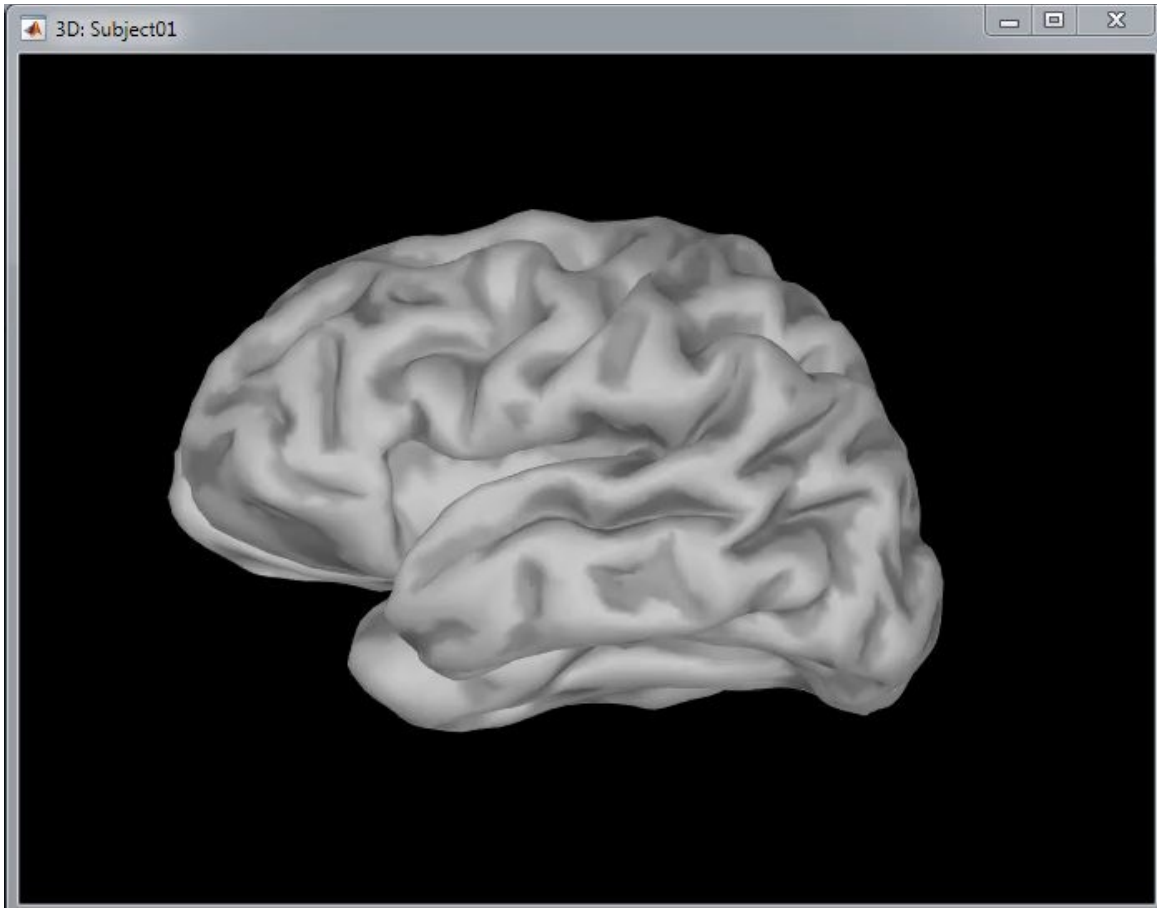


Edge Detection





Edge Detection



DEMO TIME

- Tools
- Edge Detections
- blurring
- Segmentation
- Road
- Sudoku