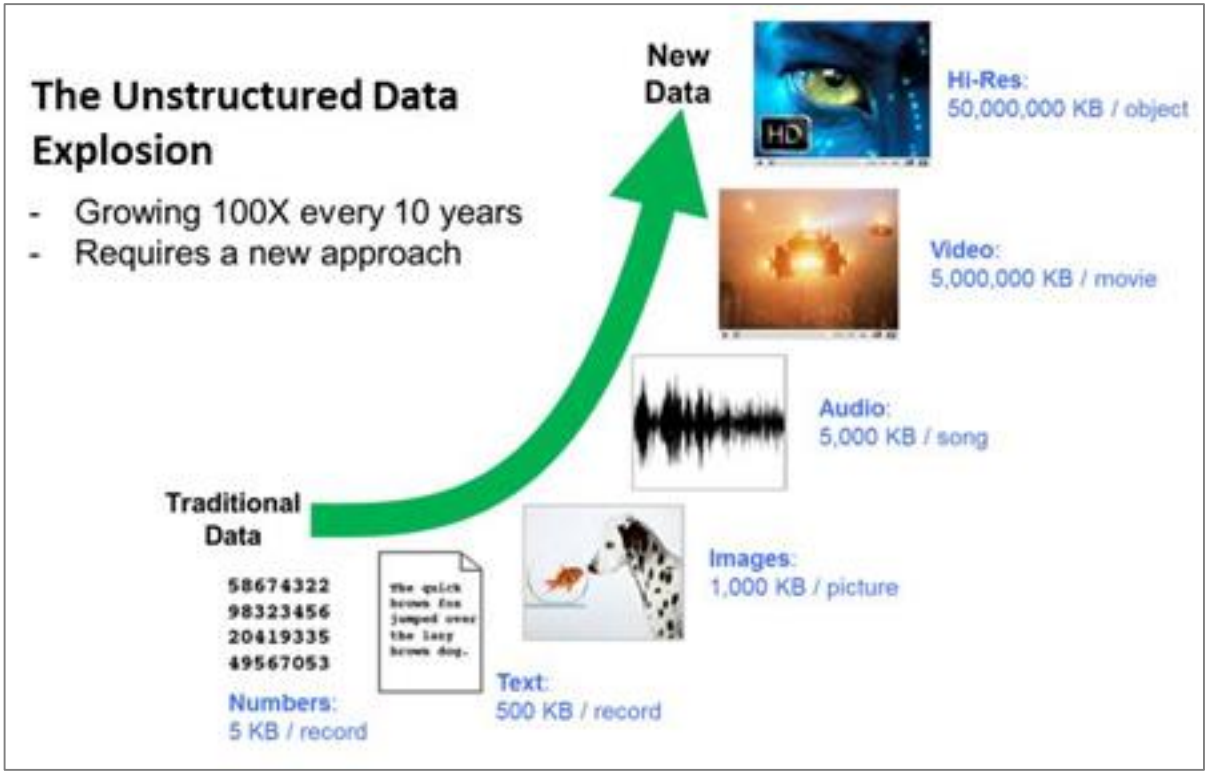
영상 데이터의 처리와 정보추출

이동윤
LG CNS
Advanced Analytics Center

빅데이터에서의 영상 데이터

- 영상 데이터를 정형 데이터처럼 다룰 필요성 ↑



http://www.datanami.com/2012/12/24/how_object_storage_and_information_dispersal_address_big_data_challenges/

강연자 소개

- 컴퓨터비젼, 패턴인식 전공 (석사)

- 산업계 경력 11년차
  - 삼성전자 반도체연구소
    - 컴퓨터비젼, 빅 데이터 (CAD)
  - LG CNS 고급분석 센터
    - 컴퓨터비젼, 고급분석/예측

※ iPad 1, iPhone 5GS 두 모델의 개발, 제조에 참여했으며 본 강의의 내용+a가 그에 적용되었음

- 페이스북 컴퓨터비젼/ 패턴인식/머신러닝 그룹 관리자 (neuralix@gmail.com)

www.facebook.com/groups/cvprml
(>2,200명)

영상 데이터로부터의 정보추출

- 컴퓨터비젼

※ 영상으로부터 '정보'를 추출하는 것을 목적으로 삼아 온 공학/과학 분과
("영상만을 보고 이해하는 컴퓨터의 실현")

어떤 종류의 장면인가?

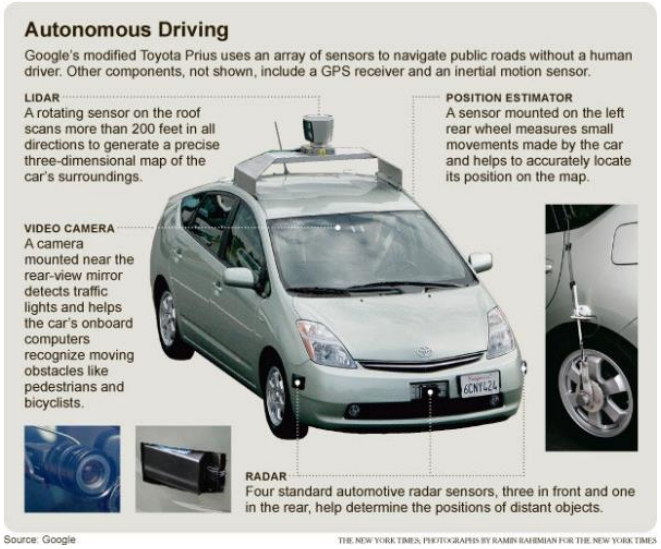
자동차가 어디에 있는가?

건물이 얼마나 멀리 있는가?

:

컴퓨터비젼 응용사례

- 구글 자동운전 자동차



※ 비디오 카메라 뿐 아니라 레이저를 이용한 LIDAR,
전파를 이용하는 RADAR에도 본 자료 내의 기술이 적용됨

http://edn.com/electronics-blogs/automotive-innovation/4402800/Self-driving-car-pushes-sensor-technology
http://www.theguardian.com/technology/2012/sep/30/google-self-driving-car-unemployment

이 강연에서...

- Data Type
  - Image (O)
  - Video (X)
- Language
  - Python (2.7.X)
- Library (패키지)
  - scikit-image

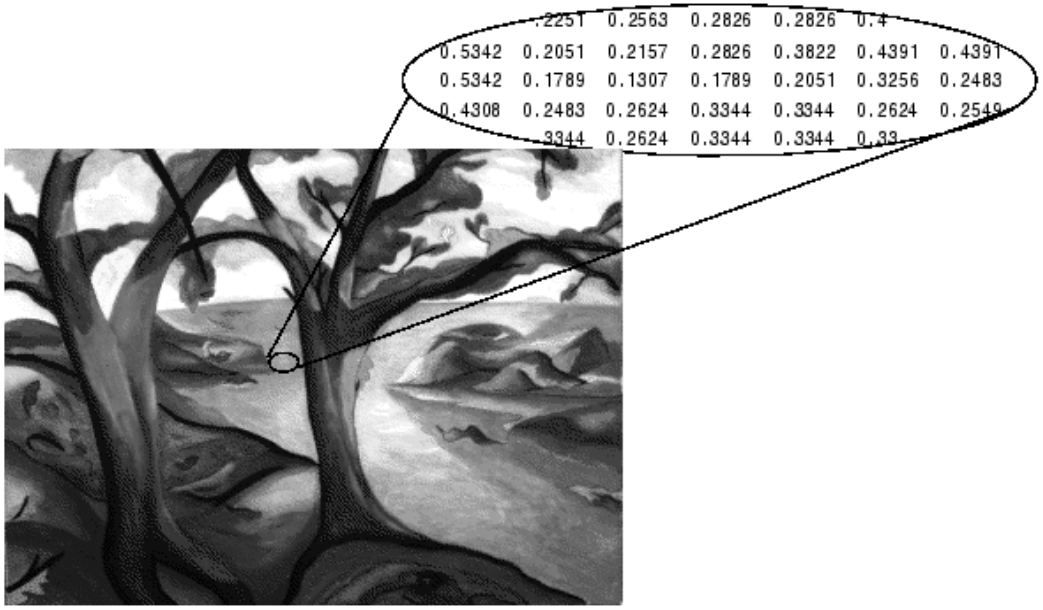
※ 컴퓨터비전 기술의 시도와 활용을 극히 쉽게 만들어줌

NOT

- GPU(CUDA), 3D, 얼굴인식, 머신러닝 관련기술, 주파수 영역분석

영상 인지의 특성

- Digitizing

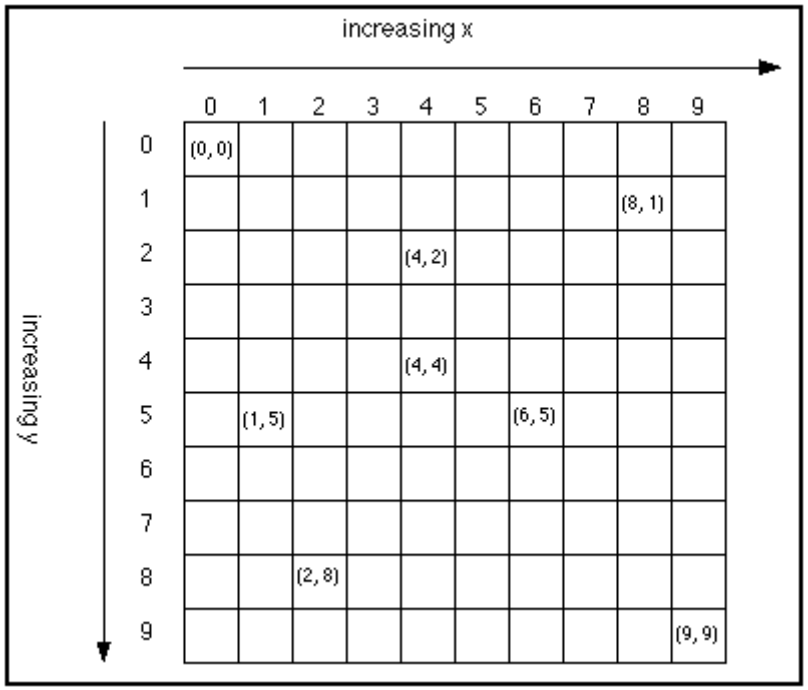


※ 가상의 격자. 이 격자로 인해 원하지 않게 문제점이 발생하기도 함

Figure Credit : Amin Allalou

영상 인지의 특성

- 영상좌표계

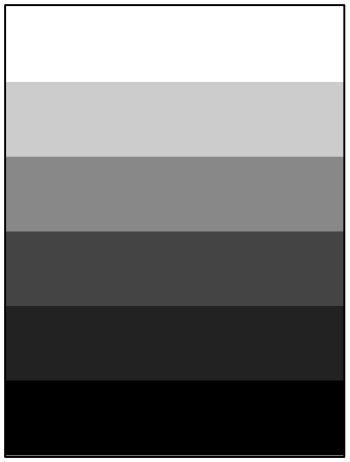


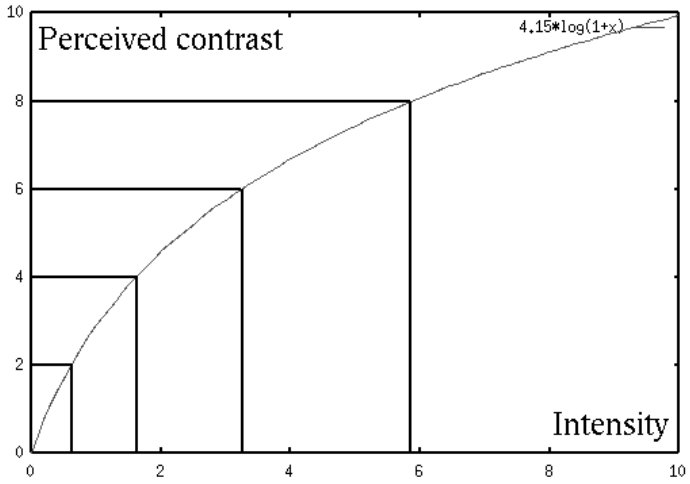
1. 좌측상단이 원점 2. 좌표계 관례 확인 필요

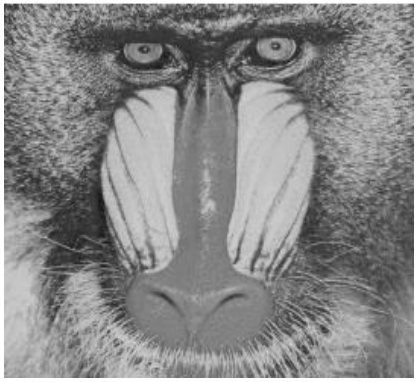
Figure Credit : Alex Lin

영상 인지의 특성

- 256 단계 픽셀







Weber-Fechner law

"한 자극의 다음 자극은 훨씬 더 커야 한다" (지수적 증가)
모든 종류의 감각자극이 이 법칙을 따름

Figure Credit : Ruye Wang

영상 파일 포맷

- 주요 영상 파일 포맷 설명

	데이터의 손실	데이터 용량	인지적 우수성	특성
BMP	無	큼 (용량고정)	N/A	색 재현성 우수, Windows OS용
TIFF				투명도 고려 가능, 가장 선호됨
PNG				위 둘보다 헤더 구조가 단순
GIF	有	작음	낮음	경계선이 부드러움, 동영상처럼 만들 수 있음(:모션 GIF)
JPG		매우 작음	매우 우수	데이터 용량이 가장 작아 네트워크에서의 전송에 유리함

차 례

- Basic Processing
- Region Processing
- Edge/Contour
- Point/Line/Corner
- Matching
- Preprocessing
- Color

※ 내용이 어려워지는 순서

Basic Processing

"이것만 하면 다 할 수 있다."
(단, 전문가라면...)

Load/Save and Display

- PIL (Python Image Library)
  - Import Module

```
>> from PIL import Image
>> from pylab import *
```

  - Load

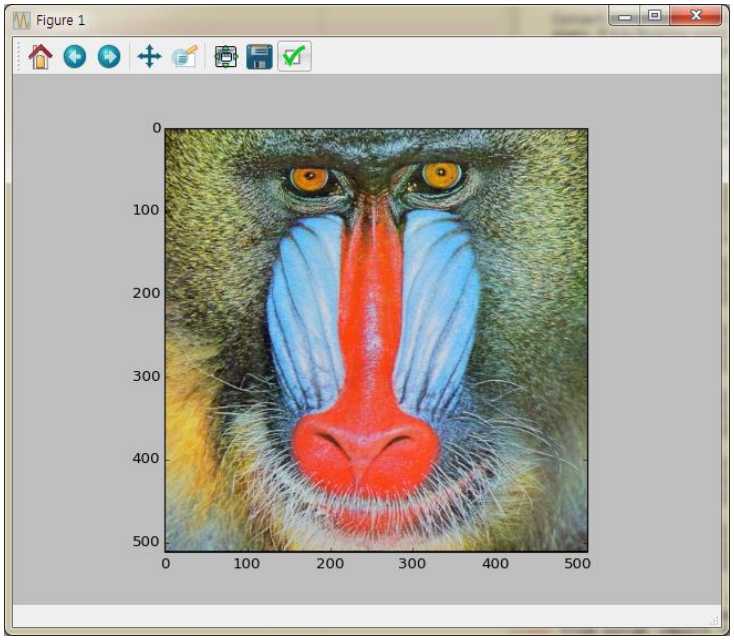
```
>> img = array(Image.open('C:₩mandrill.bmp'))
```

  - Display

```
>> imshow(img)
```

  - Save

```
>> Img = Image.fromarray(img)
>> Img.save('C:₩mandrill_new.bmp')
```



코드 안내

- 그래픽 출력용 모듈인 matplotlib와 그 서브모듈인 pylab은 임포트를 생략함
- numpy의 경우 생략
- 각 장에서, 이미 앞 부분에서 설명된 처리를 위한 모듈의 경우 임포트가 생략되었을 수 있음
- 코드 내용의 자료 기입 이후의 수정으로 일부 비작동 사례 있을 수 있음 (발견 시, neuralix@gmail.com 로 inform 요망)

Load/Save and Display

- skimage

```
>> from skimage import io
```

```
>> camera = io.imread( filename )
```

※ URL의 입력도 가능

```
>> imshow( img )
```

```
>> io.imsave( filename, logo )
```

- opencv

```
>> import numpy as np
>> import cv2
```

```
>> img = cv2.imread( filename, 0 )
```

```
>> cv2.imshow('image', img)
```

```
>> cv2.imwrite( filename, img)
```

Pixel Operation

- Get pixel value

```
>> print img[ i ][ j ][ k ]
```

- Put Pixel Value

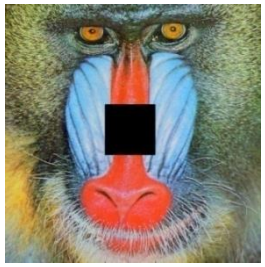
```
>> img[ i ][ j ][ k ] = 0
```

- Area Operation

```
for i in range(img.shape[0]):
  for j in range(img.shape[1]):
    for k in range(img.shape[2]):
      if i>=200 and i<300:
        if j>=200 and j<300:
          img[i][j][k] = 0 * img[i][j][k]
```

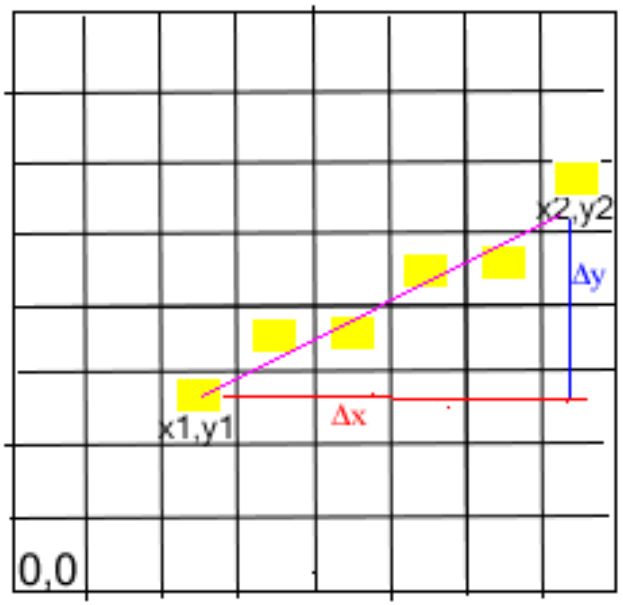
or Simply,

```
>> img[200:300][200:300] = ₩
                          0 * img[200:300][200:300]
```



Line Drawing

- DDA (Digital Differential Analyzer)



※ 가장 단순한 선긋기 방법

```
DDA(float x1, float y1 float x2, float y2) {

    float x_start, y_start;
    float xinc, yinc;
    int steps;

    /* steps는 x, y중 더 큰 차이인 쪽으로 선택한다. */
    steps = Round(x2) - Round(x1);
    xincr = (x2 - x1) / steps;
    yincr = (y2 - y1) / steps;
    x_start = x1
    y_start = y1;

    MarkPixel(Round(x_start),Round(y_start))

    for (i=0; i<steps1 i++) {
        x = x + xincr;
        y = y + yincr;
        MarkPixel(Round(x),Round(y));
     }
}
```

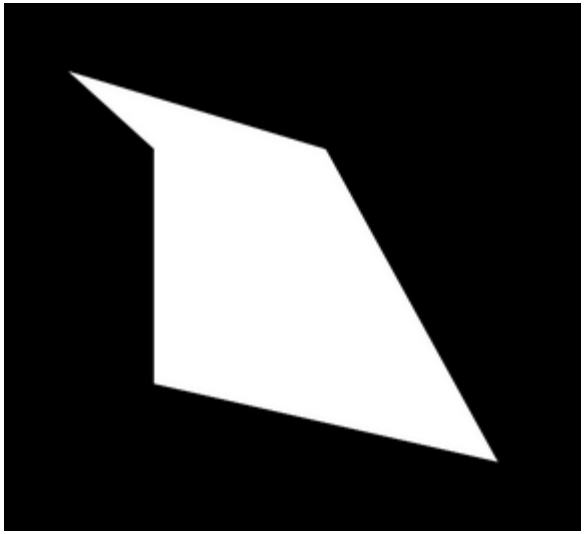
```
>> from skimage.draw import line, set_color

>> rr, cc = line(x1, y1, x2, y2)
>> set_color(img, (rr, cc), 1)
>> imshow(img)
```

Polygon Drawing

- 다각형 그리기

```
>> from skimage.draw import polygon
```

```
>> img = np.zeros((width, height), dtype=np.uint8)
>> x = np.array([x1, x2, x3, x4, …])
>> y = np.array([y1, y2, y3, y4, …])
>> rr, cc = polygon(y, x)
>> img[rr, cc] = 1
```

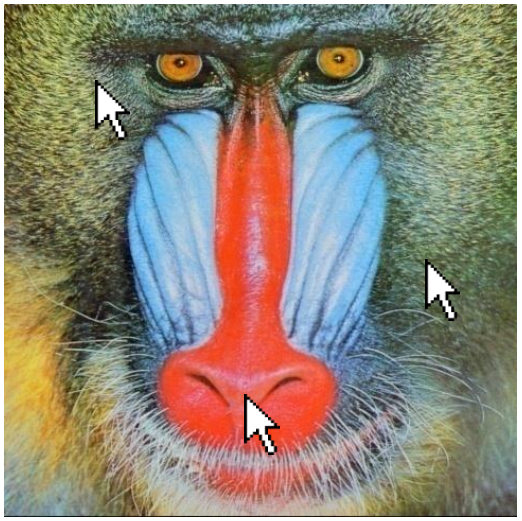


Picking Points (“ginput”)

- 클릭한 지점의 좌표를 배열로 리턴

```
>> from PIL import Image
>> from pylab import *

>> img = array(Image.open('C:₩mandrill.bmp'))
>> imshow(img)
>> print 'Please click 3 points'
>> x = ginput(3)
>> print 'you clicked:',x
>> show()
```

```
>> x
you clicked: [(16.737903225806448, 18.705645161
290249), (406.85887096774195, 23.2419354838709
18), (227.22177419354841, 242.79838709677415)]
```



※ 이름의 유래는 MATLAB의 함수 이름 (Graphical INPUT)에서.
워낙 많이 쓰이고 편리하여 보통명사처럼 됨.

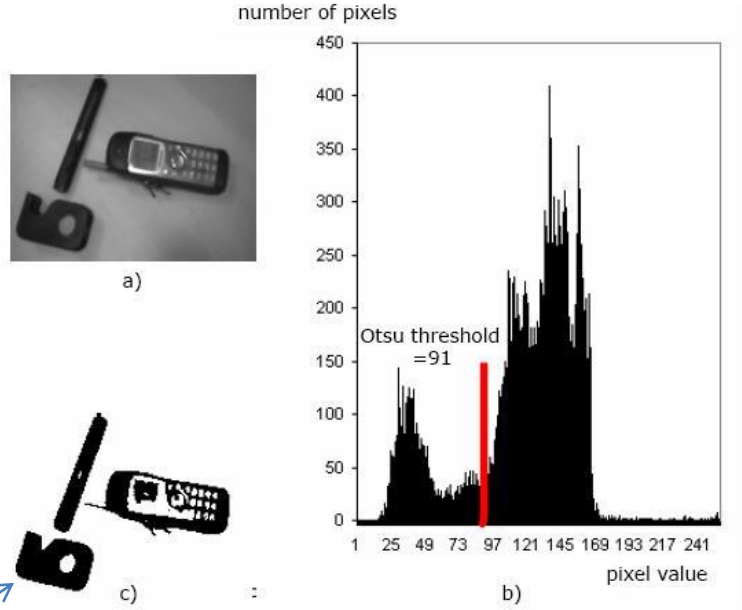
Region Processing

※ 영역의 추출과 처리

Thresholding (Binarizing)

- Otsu's (Automatic) Thresholding

```
>> from skimage import data
>> from skimage.filter import threshold_otsu, ₩
      threshold_adaptive

>> from skimage import color

>> img = io.imread('C:₩Mandrill.bmp')
>> img_g = color.rgb2grey(img)

>> val_thresh = threshold_otsu(img_g)
>> img_bw = im_g > val_thresh

>> io.imshow( img_bw )
```



"Blob"

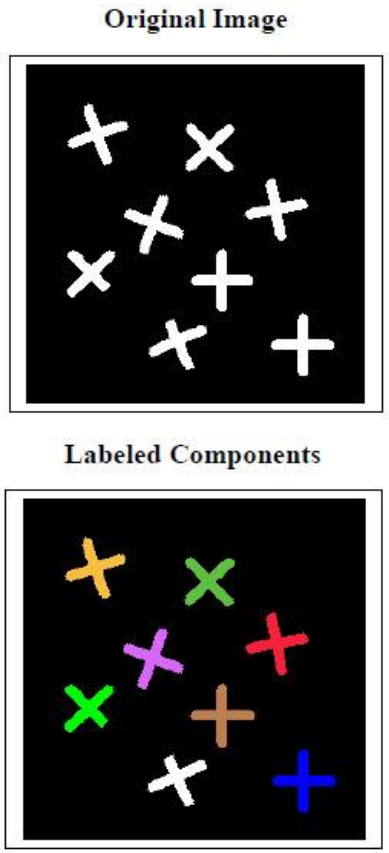
Figure Credit : https://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi_ch7/functn_ComputeThreshold_Otsu.html

Labeling

- Connected Components Labeling

```
>> from skimage.morphology import label, closing, square
>> from skimage.color import label2rgb
...
>> val_thresh = threshold_otsu(img)

>> img_label = label(val_thresh)
>> image_label_rgb = label2rgb(label_image)
>> imshow(image_label_rgb)
```

※ Check neighborhood convention first!
(8-neighborhood is complete, 4-neighbor is not.)



http://engineersphere.com/biomedical-engineering/biomedical-image-processing-iii.html

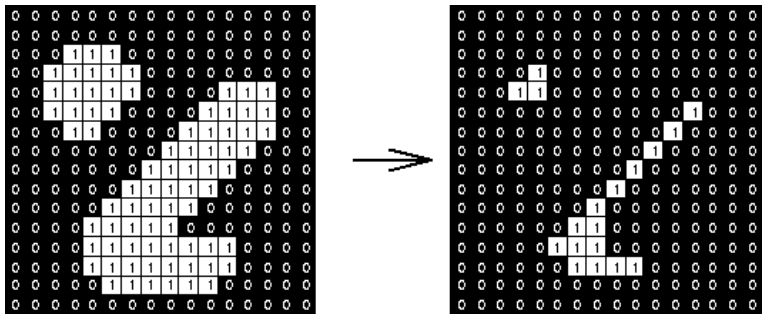
Morphological Operation

- 1) 블랍을 <뚱뚱하게/날씬하게>, 2) 서로 다른 블랍을 <붙게/떨어뜨리게>, 3) <구멍을 메우게/돌출부를 평평하게>

```
>> from skimage.morphology import erosion, dilation, opening, closing, white_tophat
>> from skimage.morphology import black_tophat, skeletonize, convex_hull_image
>> from skimage.morphology import disk
...
>> selem = disk(6)

>> img_eroded = erosion(img, selem)
>> imshow(img_eroded, cmap=plt.cm.gray)

>> img_dilated = dilation(img, selem)
>> imshow(img_dilated, cmap=plt.cm.gray)
```

※ Structuring element (:selem)의 모양과 크기에 의해 조정됨

※ LIDAR/RADAR 데이터의 처리에서 중요



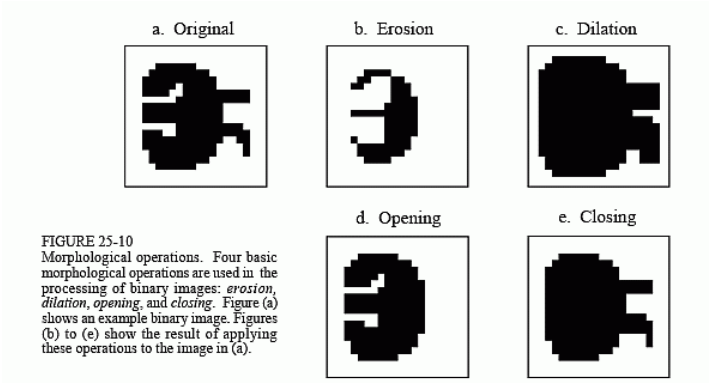


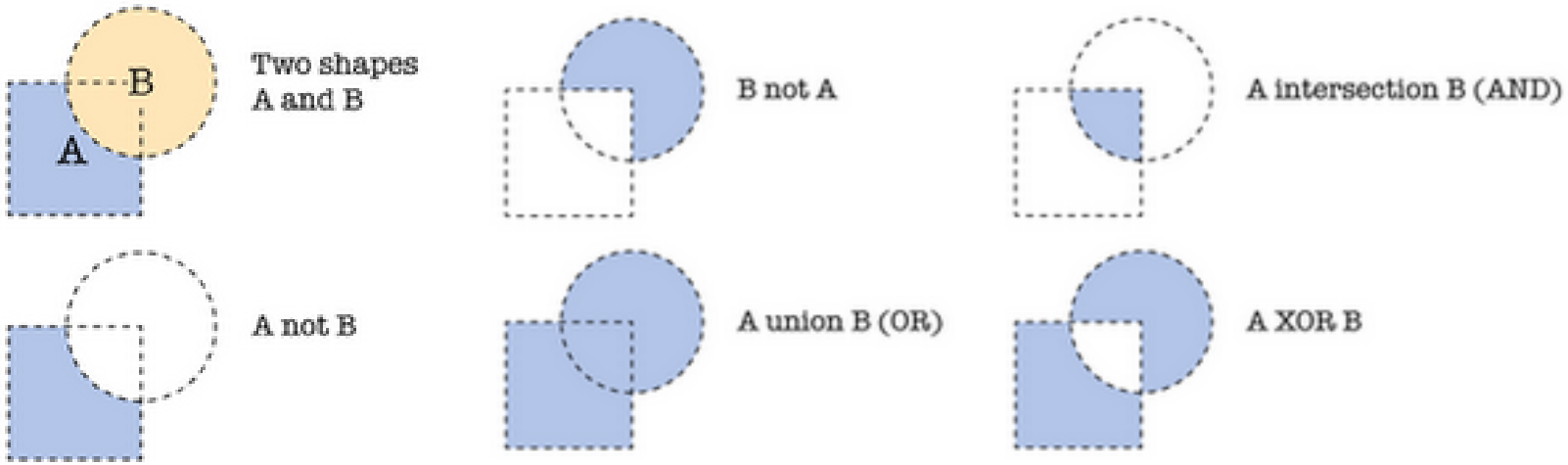
FIGURE 25-10
Morphological operations. Four basic morphological operations are used in the processing of binary images: *erosion*, *dilation*, *opening*, and *closing*. Figure (a) shows an example binary image. Figures (b) to (e) show the result of applying these operations to the image in (a).

https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

http://www.dspguide.com/ch25/4.htm

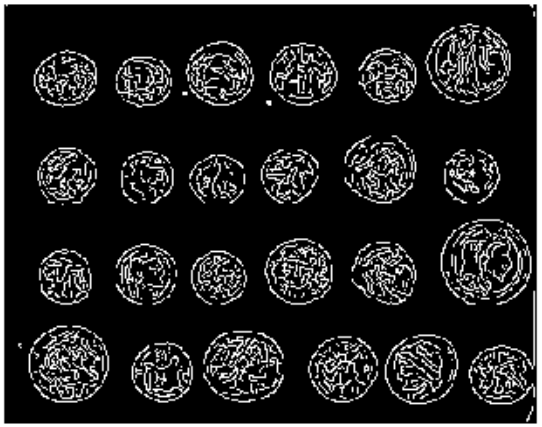
Boolean Operation

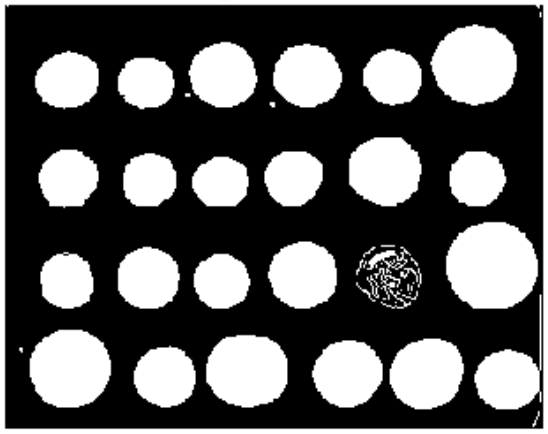
- Image Boolean Operations



Flood Fill

- Filling the holes






```
>> from scipy import ndimage
>> from skimage.filter import canny

>> img_edges = canny(img/255.)
>> img_filled = ndimage.binary_fill_holes(img_edges)
```

Segmentation

- Mean Shift

```
>> import cv2
>> import pymeanshift as pms

>> img_orig = cv2.imread("example.png")

>>(img_segmented, img_label, number_regions) = ₩
        pms.segment(img_orig, spatial_radius=6, ₩
                        range_radius=4.5, min_density=50)
```

※ 구획화(Segmentation)는 경계를 인지, 이용하는 방향과 내부 유사성을 인지, 이용하는 두 계열이 있음. GraphCuts, GrabCuts는 전자, Mean Shift(:Moving Averaging)은 후자, Chan's Active Contour without Edges는 둘을 모두 추구함

https://code.google.com/p/pymeanshift/

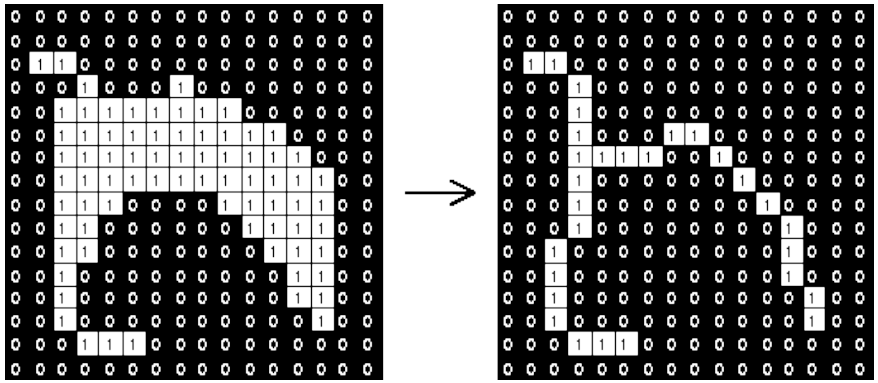
Edge / Contour

※ 임의의 선, 곡선들

Skeletonization

- Skeletonization

```
>> from skimage.morphology import skeletonize
>> img_skeleton = skeletonize(image)
>> imshow(img_skeleton, cmap=plt.cm.gray)
```

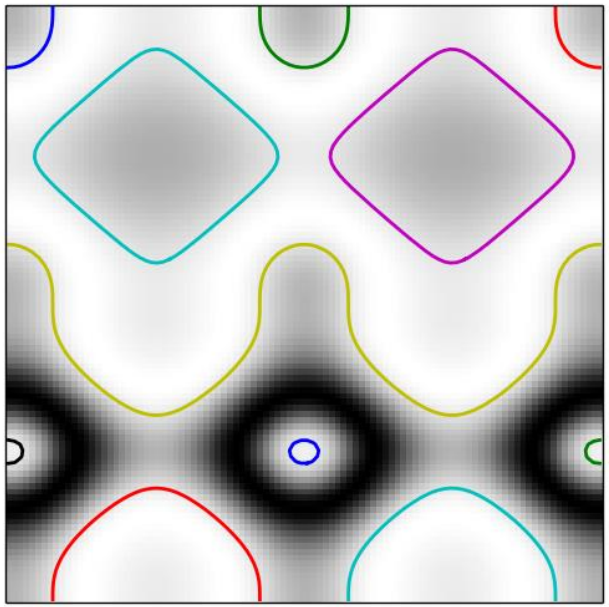


http://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm

Contour Finding

- Contour Finding

```
>> import numpy as np
>> import matplotlib.pyplot as plt
>> from skimage import measure

>> x, y = np.ogrid[-np.pi:np.pi:100j, -np.pi:np.pi:100j]
>> r = np.sin(np.exp((np.sin(x)**3 + np.cos(y)**2)))

>> contours = measure.find_contours(r, 0.8)

>> fig, ax = plt.subplots()
>> ax.imshow(r, interpolation='nearest', cmap=plt.cm.gray)

>> for n, contour in enumerate(contours):
        ax.plot(contour[:, 1], contour[:, 0], linewidth=2)

>> plt.show()
```



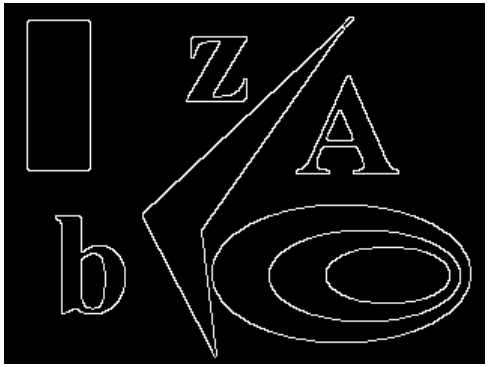
※ 콘투어의 각 점들을 배열로 추출해줌

Edge Detection (1of2)

- Canny Edge Detection

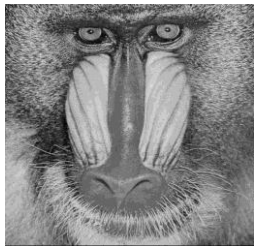
```
>> import numpy as np
>> import matplotlib.pyplot as plt
>> from scipy import ndimage

>> from skimage import filter

>> img_edges = filter.canny(img, sigma=3)
>> imshow(img_edges, cmap=plt.cm.gray)
```

※ 콘투어가 전체에서 동일한 값인 지점을 주는 것이라면 에지는 국소적인 조사로 최적경계를 찾으므로 대개의 경우에 더 효과적임 (1차미분과 유사함).
※ 위의 경우는 Thresholding 후 에지 검출이 가능

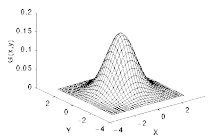


Edge Detection (2of2)

- Canny Edge Detection ※ 에지 검출 기법의 대명사와 같음 (“Canny”)



*



$$\text{Edge\_Gradient } (G) = \sqrt{G_x^2 + G_y^2}$$

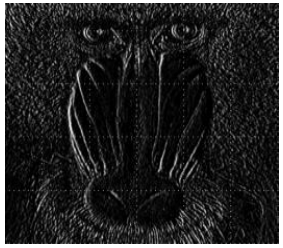
$$\text{Angle } (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

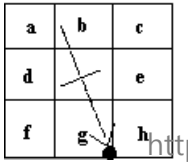
*

-1 0 1
-1 0 1
-1 0 1

*

-1 -1 -1
0 0 0
1 1 1





http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

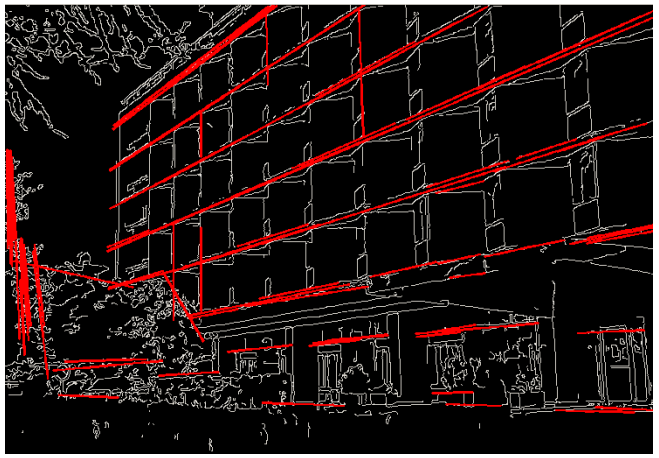
http://news.mynavi.jp/photo/series/computer_vision/040/images/014l.jpg

http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/low/edges/canny.htm

Point / Line / Corner

Hough Transform (1of2)

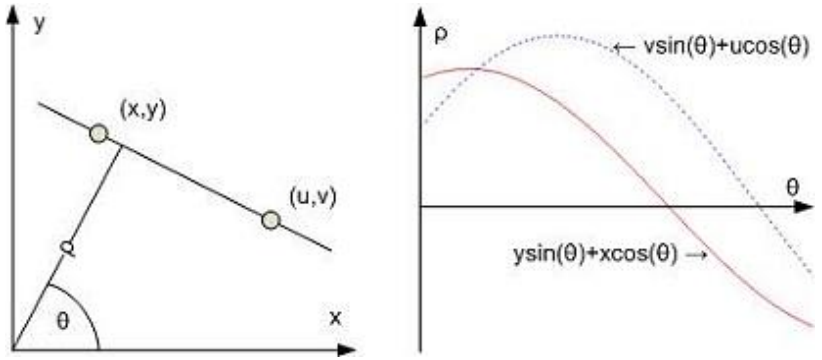
- Hough Transform



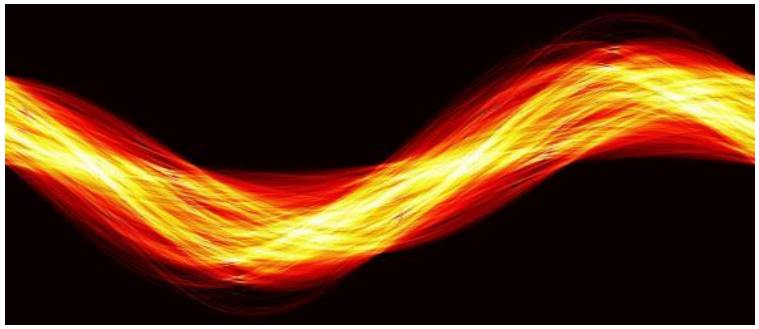
http://docs.opencv.org/modules/imgproc/doc/feature_detection.html

Hough Transform (2of2)

- Hough Transform




```
>>from skimage.transform import (hough_line, hough_l
ine_peaks, probabilistic_hough_line)

>> edges = canny(image, 2, 1, 25)
>> lines = probabilistic_hough_line(edges, threshold=₩
                                10, line_length=5, line_gap=3)

>> for line in lines:
        p0, p1 = line
        plt.plot((p0[0], p1[0]), (p0[1], p1[1]))

>> plt.show()
```



※ rho, theta 평면의 한 지점은 x, y 평면의 하나의 직선에 대응함

※ 두 점을 잇는 모든 직선을 rho, theta 평면에서 voting하여 피크지점을 추출하여 직선을 추정함

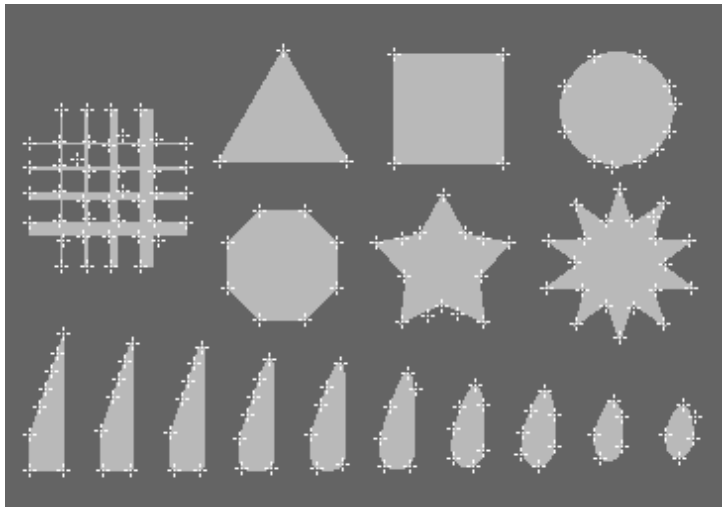
http://campar.in.tum.de/Students/DaPentenrieder

http://www.aiaa-daycin.org/node/430

Corner Detection (1of2)

- Harris Corner Detection

```
>> from skimage.feature import corner_harris, corner
_subpix, corner_peaks

>> coords = corner_peaks(corner_harris(image), min_
distance=5)
>> coords_subpix = corner_subpix(image, coords, win
dow_size=13)

>> fig, ax = plt.subplots()
>> ax.imshow(image, interpolation='nearest', cmap=p
lt.cm.gray)
>> ax.plot(coords[:, 1], coords[:, 0], '.b', markersize=3)
>> ax.plot(coords_subpix[:, 1], coords_subpix[:, 0], '+r',
markersize=15)
>> ax.axis((0, 350, 350, 0))

>> plt.show()
```



$$\mathbf{M} = \begin{bmatrix} \left(\frac{\partial f}{\partial x}\right)^2 & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \left(\frac{\partial f}{\partial y}\right)^2 \end{bmatrix}$$

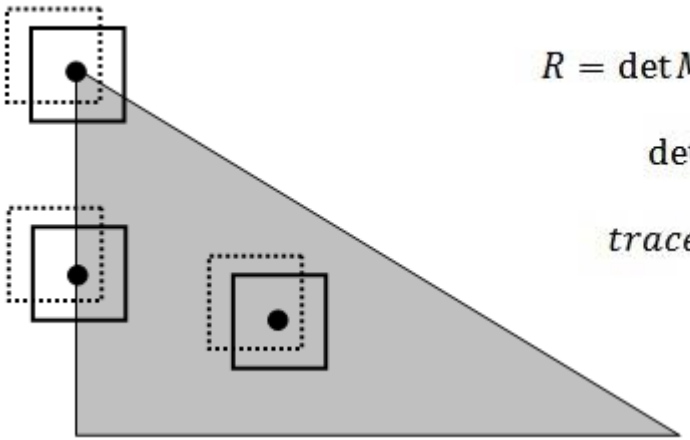
$$R = det\,\mathbf{M} - k\,Trace^2\,\mathbf{M}$$

https://www.ee.iitb.ac.in/~sumantra/courses/ip/assignment_1.html

http://www.krankerkaktus.de/Portfolio/GraphicsProgramming

Corner Detection (2of2)

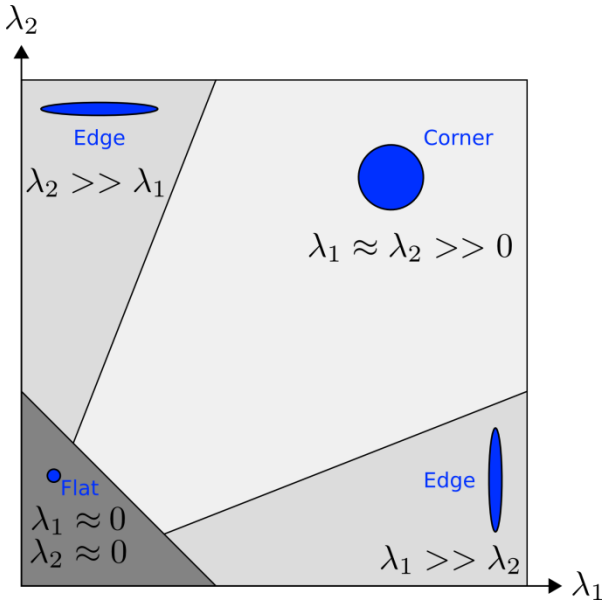
- Harris Corner Detection



$$R = \det M - k(trace\, M)^2$$

$$\det M = \lambda_1 \lambda_2$$

$$trace\, M = \lambda_1 + \lambda_2$$



※ 사각형 영역을 조사해보아 주성분분석을 하였을 때, 최대값인 아이겐밸류 lambda 1, 2가 모두 0보다 꽤 크고, 서로간에 값이 비슷하면 Corner라 판단해도 좋다.

http://www.mathworks.co.kr/kr/help/images/detect-corners-in-images.html

http://miac.unibas.ch/BIA/05-LandmarkBasedRegistration.html

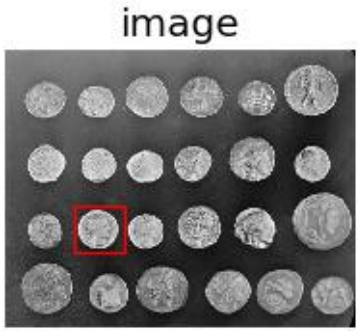
Matching / Registration

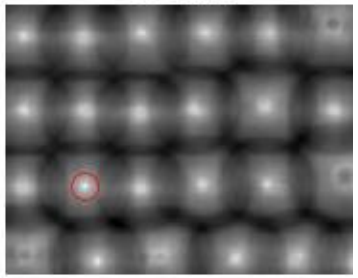
Image Matching

- Template Matching

```
>>import numpy as np
>>import matplotlib.pyplot as plt

>>from skimage import data
>>from skimage.feature import match_template
```

```
>>result = match_template( img_whole, img_template )
```

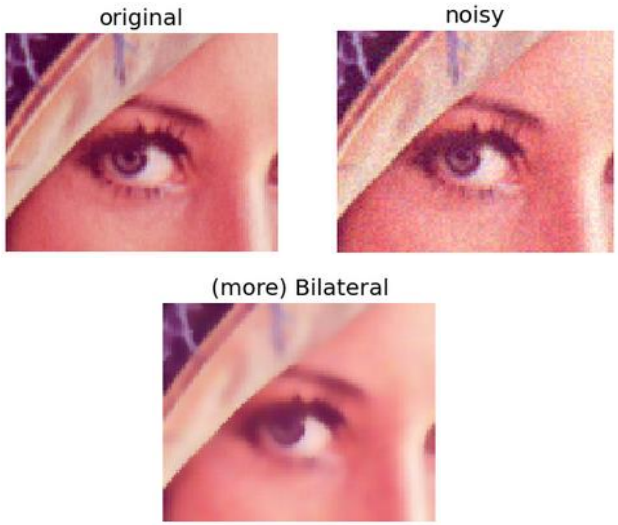




Noise Suppression

- Denoising

```
>> from skimage import data, img_as_float
>> from skimage.restoration import denoise_tv_ch
ambolle, denoise_bilateral
…
>> imshow(denoise_bilateral(img, sigma_range=0.1,
sigma_spatial=15))
```



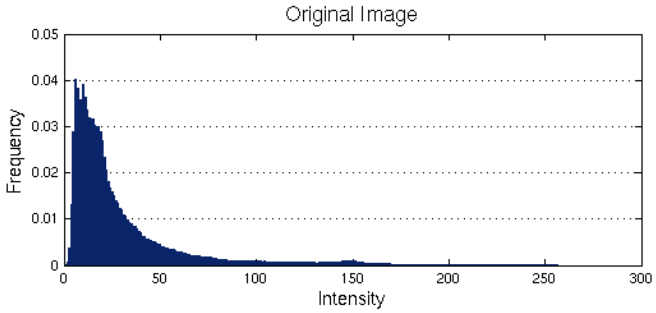
- Median

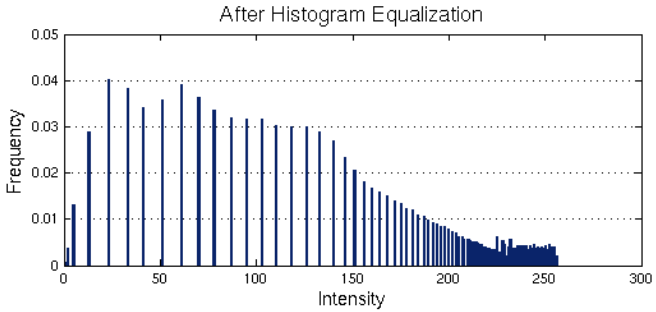
```
>> from skimage.morphology import disk
>> from skimage.filter.rank import median
…
>> img_med = median(img, disk(5))
```

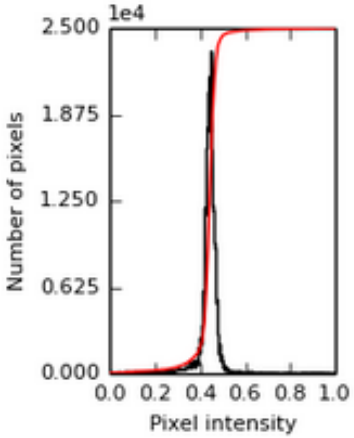
Contrast

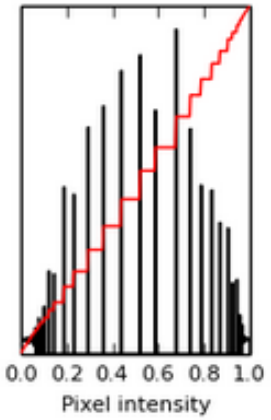
- Histogram Equalizing

```
>> from skimage import exposure
...
>> img_eq = exposure.equalize_hist(img)
```







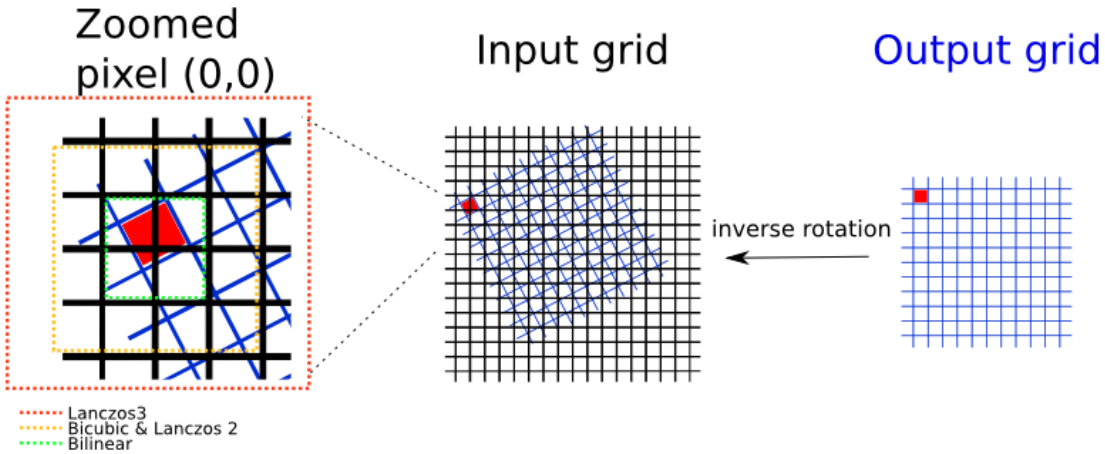


※ 히스토그램 변환규칙을 만들고 그 히스토그램을 참고해 모든 픽셀의 값을 변경함. 여러 기법 중 히스토그램 이퀄라이징은 CDF가 강제로 직선이도록 변환규칙을 만들고 이를 따르도록 하여 매 영상마다 최적에 가깝게 콘트라스트를 개선함.

Figure Credit : James Fishbaugh

Rotation

- Rotation with Interpolation



※ 모든 rotation과 resizing은 격자 불일치를 초래해 interpolation에 의한 보정을 필요로 함. Skimage는 이의 정밀한 보정에서 아직 빈약하여 opencv를 권장. 이때 image.shape,flags 변수의 적절한 설정이 필요함.

```
>> from skimage.transform import rotate
...
>> img = rotate(img, 90, resize=True).shape
```

```
>> import numpy as np
...
>> coord_center = tuple(np.array(image.shape)/2)
>> rot_mat = cv2.getRotationMatrix2D(coord_center,angle,1.0)
>> result = cv2.warpAffine(image, rot_mat, image.shape,flags=cv2.INTER_LINEAR)
```

http://www.darktable.org/2012/06/upcoming-features-new-interpolation-modes-and-better-resize/dt-rotation-grid/

Color (1of3)

- RGB → Grey

Original
(Color)

Luminousity
(Grey Value)

0.21 R + 0.72 G + 0.07 B

Average

(R + G + B) / 3

Lightness

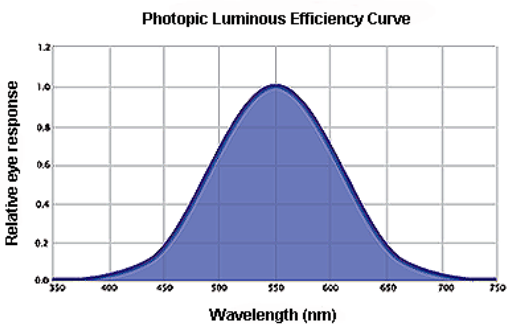
(max(R, G, B)
+ min(R, G, B)) / 2

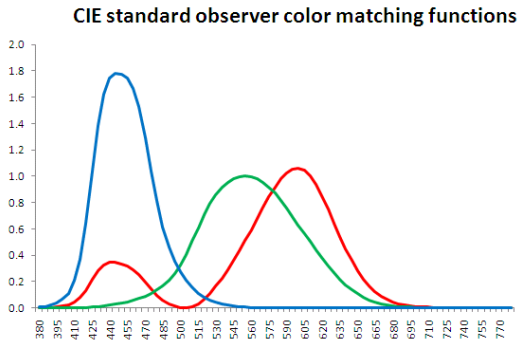
```
>> from skimage.color import rgb2gray
>> from skimage import data
...
>> img_gray = rgb2gray(img)       // Luminousity
```

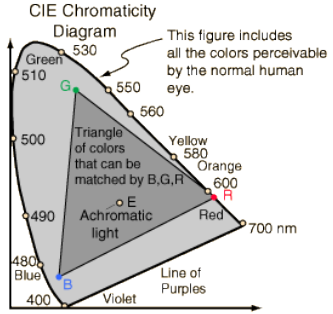
Figure Credit : Hohn D. Cook (http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/)

Color (2of3)

- Radiometry → Photometry → Colorimetry





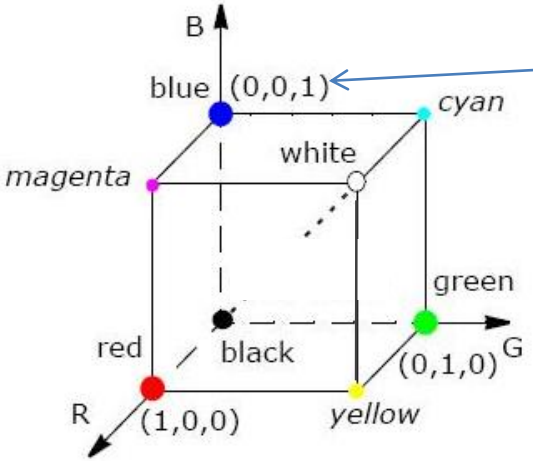


http://www.chromapure.com/products-d3pro.asp

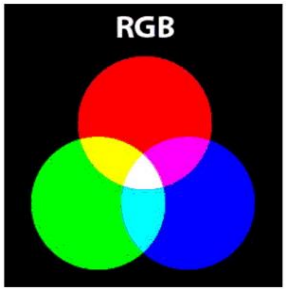
http://www.screentekinc.com/resource-center/photometry-and-colorimetry.shtml

Color (3of3)

- Color Space



※ 슬라이드에서는 1이 255에 해당



※ RGB와 CMY는 반대
(Cyan: R=0, G=255, B=255)

http://en.wikipedia.org/wiki/File:RGB_sliders.svg

https://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi_ch6/ch6_color_models.html

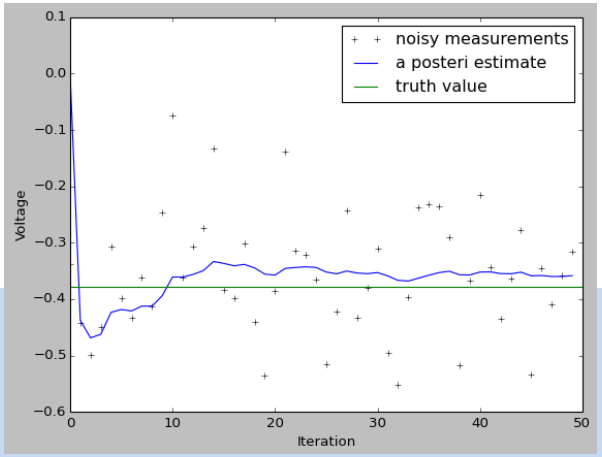
Kalman Filter
(linear quadratic estimation)

Kalman Filtering

- Kalman Filtering

※ 시계열 데이터의 예측과 평활화에 적용

※ "뉴튼 물리학과 칼만 필터로 인간을 달에 보냈다." (NASA)
"Stanley는 모든게 칼만 필터였다." (세바스찬 스런)




```
# Kalman filter example demo in Python

# A Python implementation of the example given in pages 11-15 of "An
# Introduction to the Kalman Filter" by Greg Welch and Gary Bishop,
# University of North Carolina at Chapel Hill, Department of Computer
# Science, TR 95-041,
# http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html

# by Andrew D. Straw

import numpy
import pylab

# intial parameters
n_iter = 50
sz = (n_iter,) # size of array
x = -0.37727 # truth value (typo in example at top of p. 13 calls this z)
z = numpy.random.normal(x,0.1,size=sz) # observations (normal about x, sigma=0.1)

Q = 1e-5 # process variance

# allocate space for arrays
xhat=numpy.zeros(sz)      # a posteri estimate of x
P=numpy.zeros(sz)         # a posteri error estimate
xhatminus=numpy.zeros(sz) # a priori estimate of x
Pminus=numpy.zeros(sz)    # a priori error estimate
K=numpy.zeros(sz)         # gain or blending factor

R = 0.1**2 # estimate of measurement variance, change to see effect
```

※ z : 노이즈가 가미된 입력
xhat : 추정된 출력

```
# intial guesses
xhat[0] = 0.0
P[0] = 1.0

for k in range(1,n_iter):
    # time update
    xhatminus[k] = xhat[k-1]
    Pminus[k] = P[k-1]+Q

    # measurement update
    K[k] = Pminus[k]/( Pminus[k]+R )
    xhat[k] = xhatminus[k]+K[k]*(z[k]-xhatminus[k])
    P[k] = (1-K[k])*Pminus[k]

pylab.figure()
pylab.plot(z,'k+',label='noisy measurements')
pylab.plot(xhat,'b-',label='a posteri estimate')
pylab.axhline(x,color='g',label='truth value')
pylab.legend()
pylab.xlabel('Iteration')
pylab.ylabel('Voltage')

pylab.figure()
valid_iter = range(1,n_iter) # Pminus not valid at step 0
pylab.plot(valid_iter,Pminus[valid_iter],label='a priori error estimate')
pylab.xlabel('Iteration')
pylab.ylabel('$(Voltage)^2$')
pylab.setp(pylab.gca(),'ylim',[0,.01])
pylab.show()
```

http://wiki.scipy.org/Cookbook/KalmanFiltering

부가자료

- 파이썬용 Package, Tools
  - Anaconda Distribution https://store.continuum.io/cshop/anaconda
    ※ opencv, scikit-image, spyder를 모두 담음
  - Scikit-Image Package
    - http://scikit-image.org
    - http://scikit-image.org/docs/dev/auto_examples
  - Pycharm IDE http://www.jetbrains.com/pycharm
  - 기타 Spyder IDE, iPython QTConsole, Sublime Text 2 등
- **책 (국내 및 번역서)**
  - http://book.naver.com/bookdb/book_detail.nhn?bid=6191572
  - http://book.naver.com/bookdb/book_detail.nhn?bid=6802764
  - http://book.naver.com/bookdb/book_detail.nhn?bid=6332184
  - http://kangcom.com/sub/view.asp?sku=200311040002 (컬러)
- **책 (원서)**
  - http://www.amazon.com/Machine-Vision-Ramesh-Jain/dp/0070320187
  - http://www.amazon.com/Computer-Vision-Algorithms-Applications-Science/dp/1848829345/ref=sr_1_1?s=books&ie=UTF8&qid=1401379429&sr=1-1&keywords=szeliski+computer+vision

끝