

# Lecture 07

# Filters & Features

Song Chaoyang

Assistant Professor

Department of Mechanical and Energy Engineering

[songcy@sustc.edu.cn](mailto:songcy@sustc.edu.cn)

# Agenda

---

- Filters
  - Three Views of Filters
  - Linear Filters
- Feature Extraction
  - Definition
  - OpenCV
- Point Cloud Processing: PCL

# Image Filtering

## Three Views of Understanding

- Image filters in spatial domain
  - Filter is a mathematical operation of a grid of numbers
  - Smoothing, sharpening, measuring texture
- Image filters in the frequency domain
  - Filtering is a way to modify the frequencies of images
  - Denoising, sampling, image compression
- Templates and Image Pyramids
  - Filtering is a way to match a template to the image
  - Detection, coarse-to-fine registration

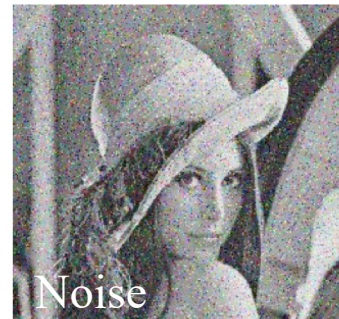
*Images are usually preprocessed by filters before feature detection*



Original Image



Distorted image 1



Noise



Contrast

lenna\_color.gif

512x512

lenna\_gray.gif

512x512

# Image Filtering

## Math View

- Compute function of local neighborhood at each position
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
  - Deep Convolutional Networks

$g[\cdot, \cdot]$

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

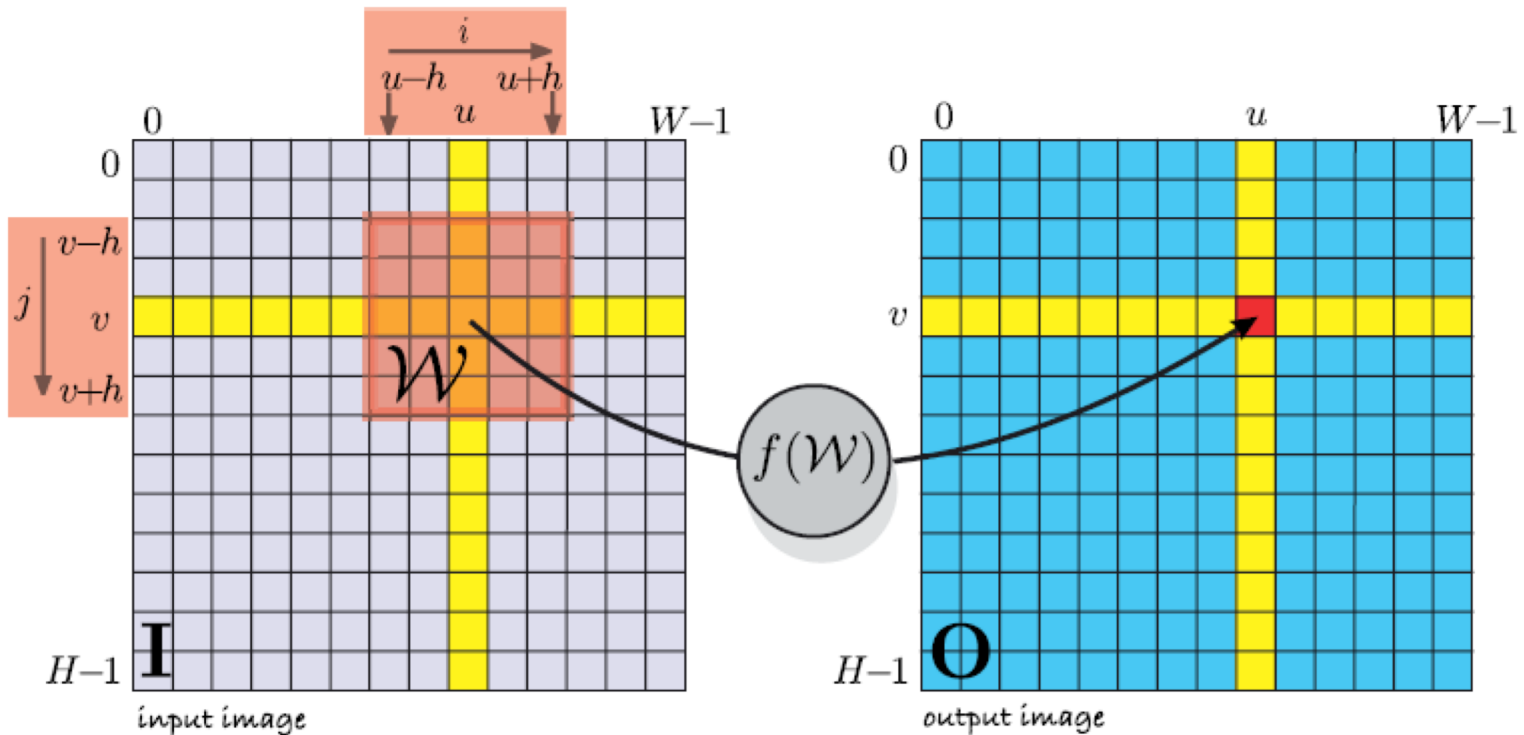
Example: box filter



# Linear Spatial Operations

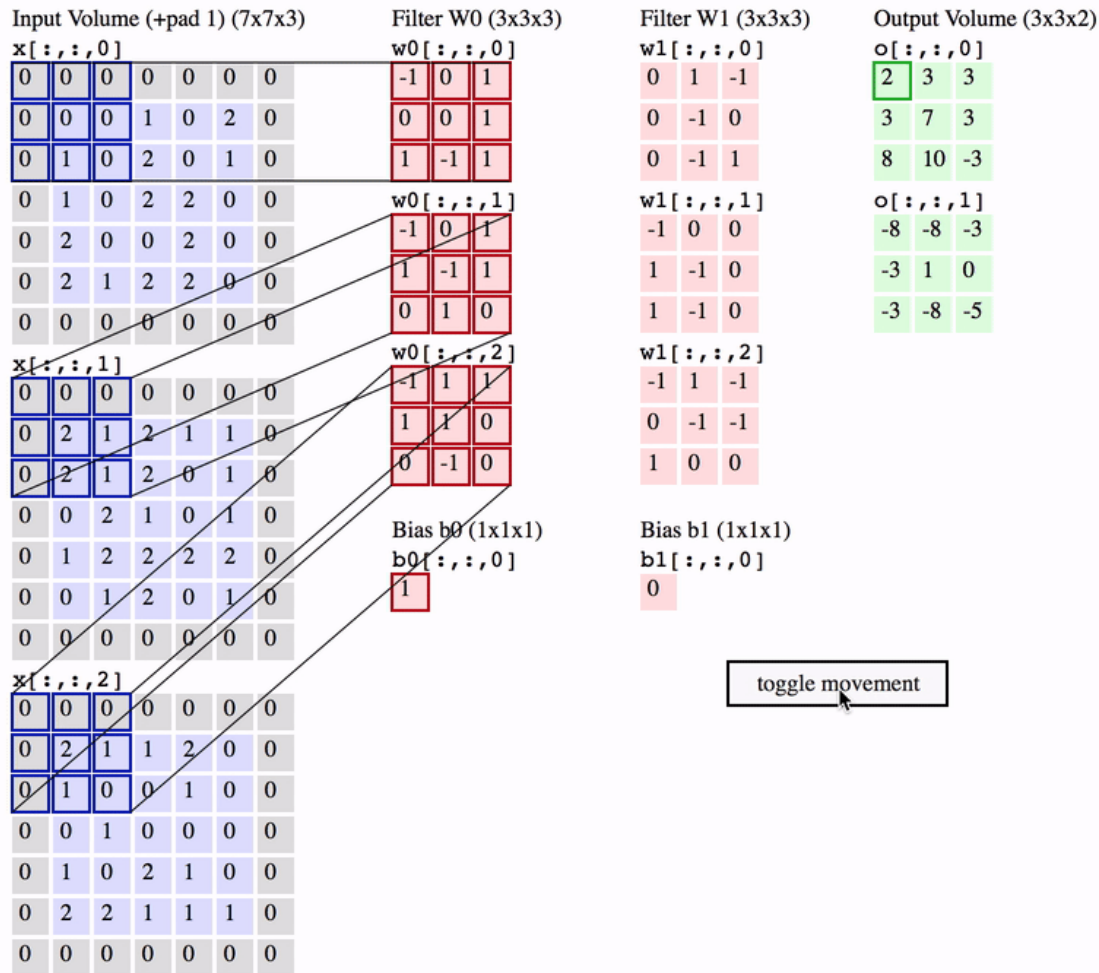
## Convolution

$$O[u, v] = \sum_{(i, j) \in \mathcal{W}} I[u + i, v + j] K[i, j], \quad \forall (u, v) \in \mathcal{I}$$



# Linear spatial operations

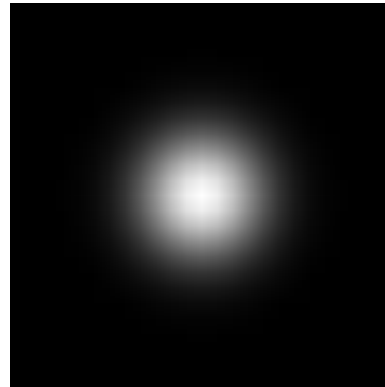
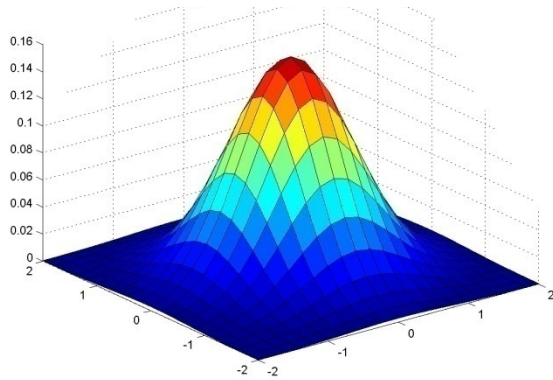
## Convolution



# Examples of Linear Filters

## Gaussian Filter

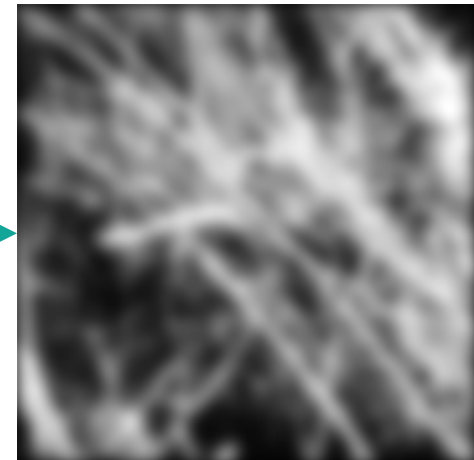
- Weight contributions of neighboring pixels by nearness



0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

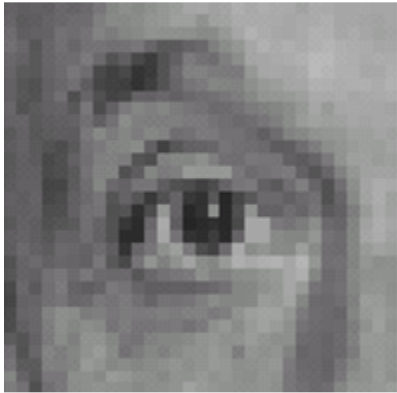


$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

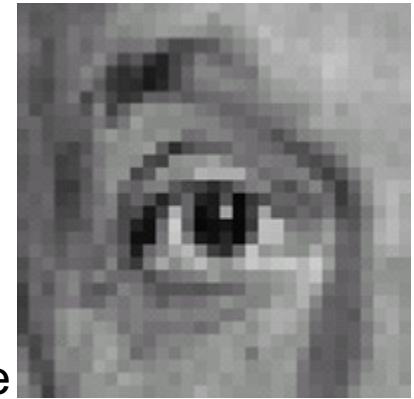


# Examples of Linear Filters

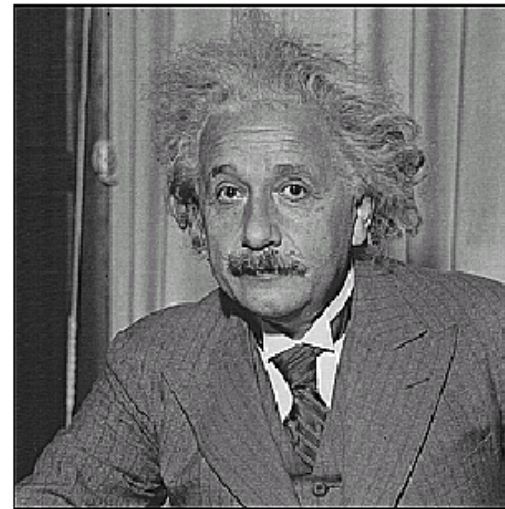
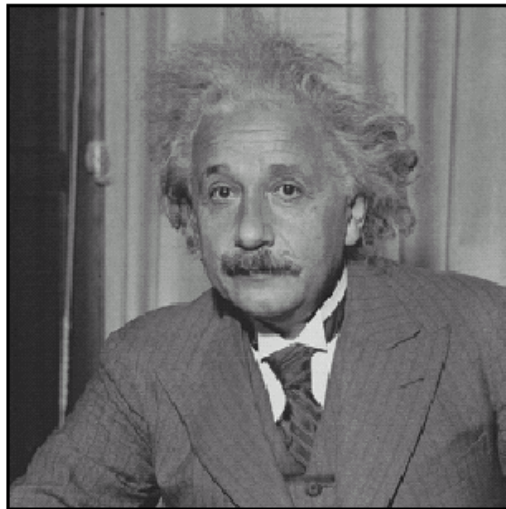
## Sharpening Filter



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

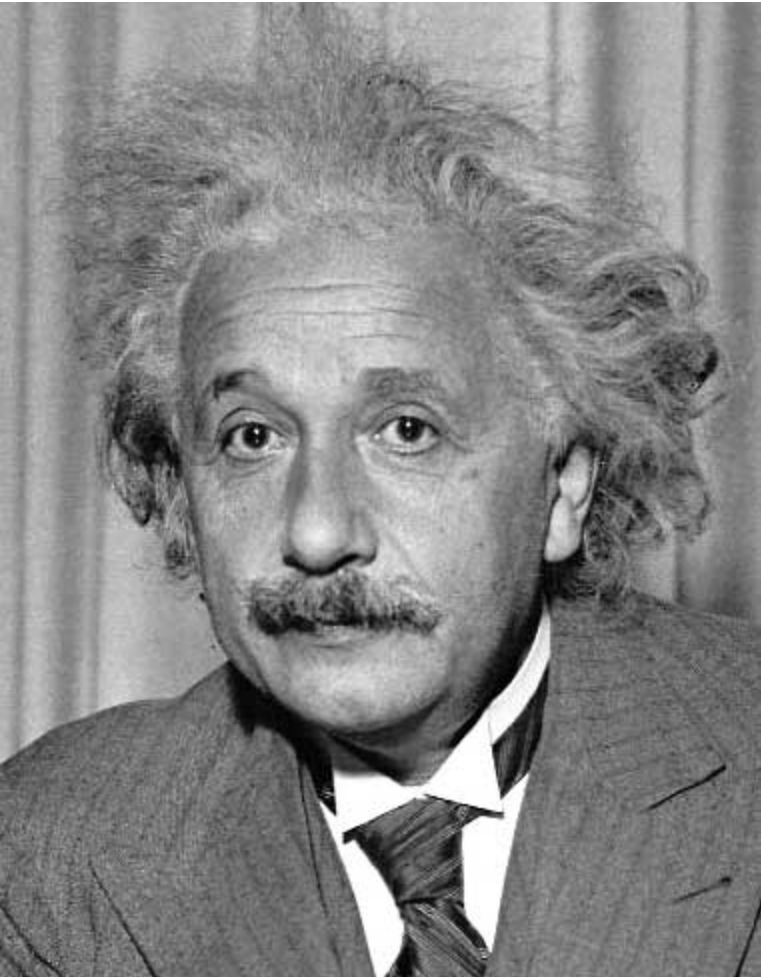


Accentuates differences with local average



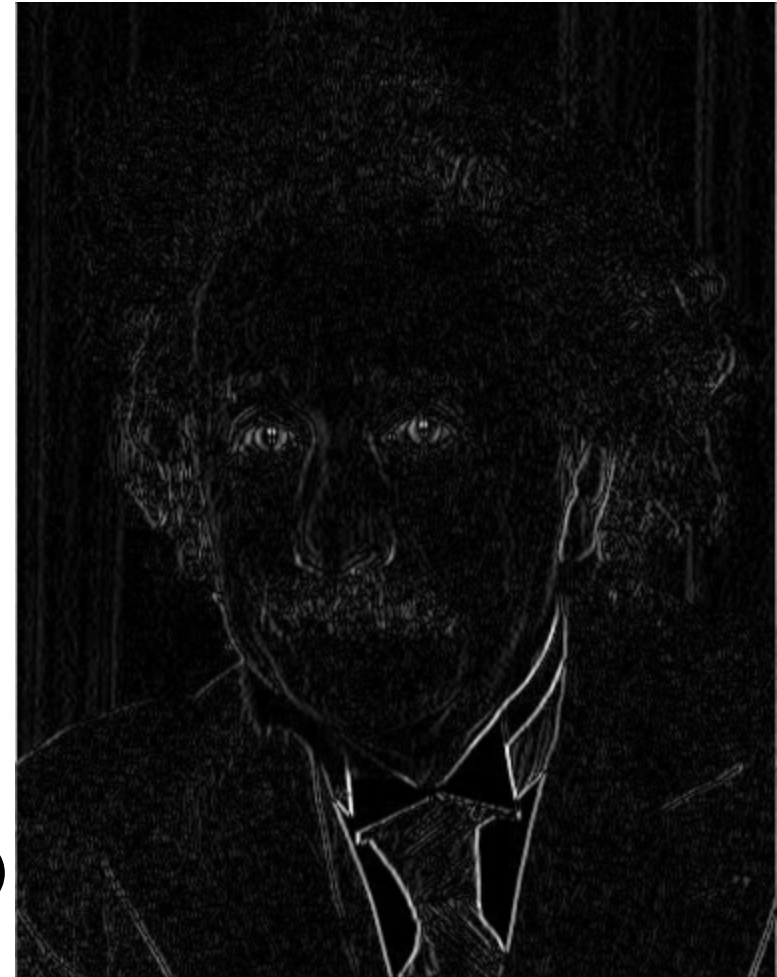
# Examples of Linear Filters

## Sobel kernel



1	0	-1
2	0	-2
1	0	-1

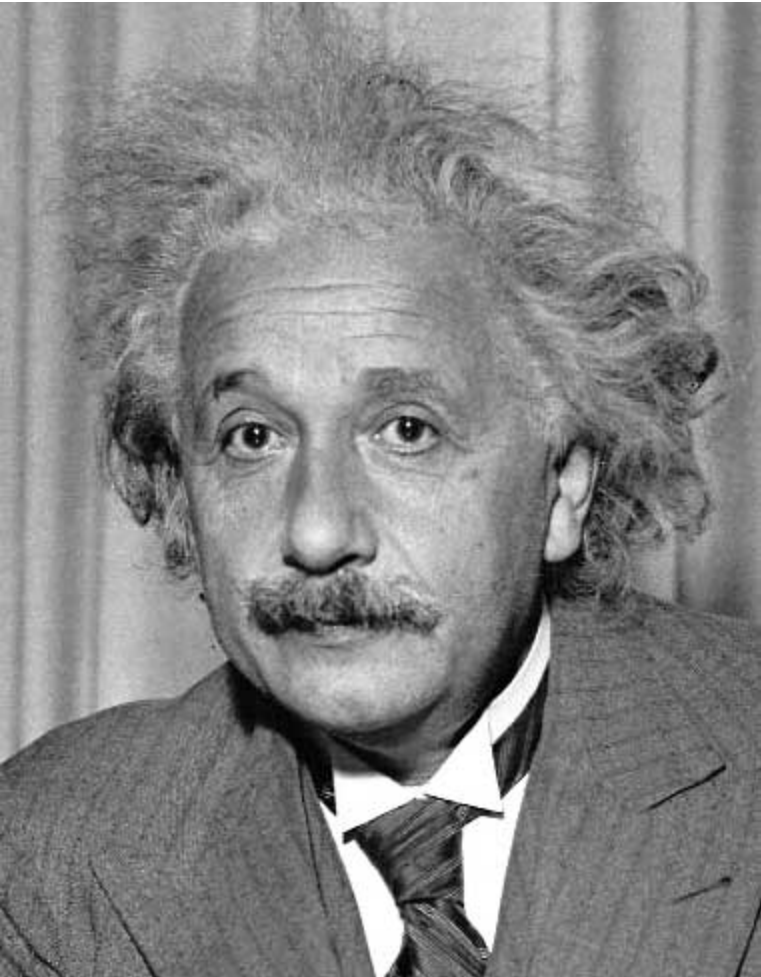
Vertical Edge  
(absolute value)





# Examples of Linear Filters

## Sobel kernel



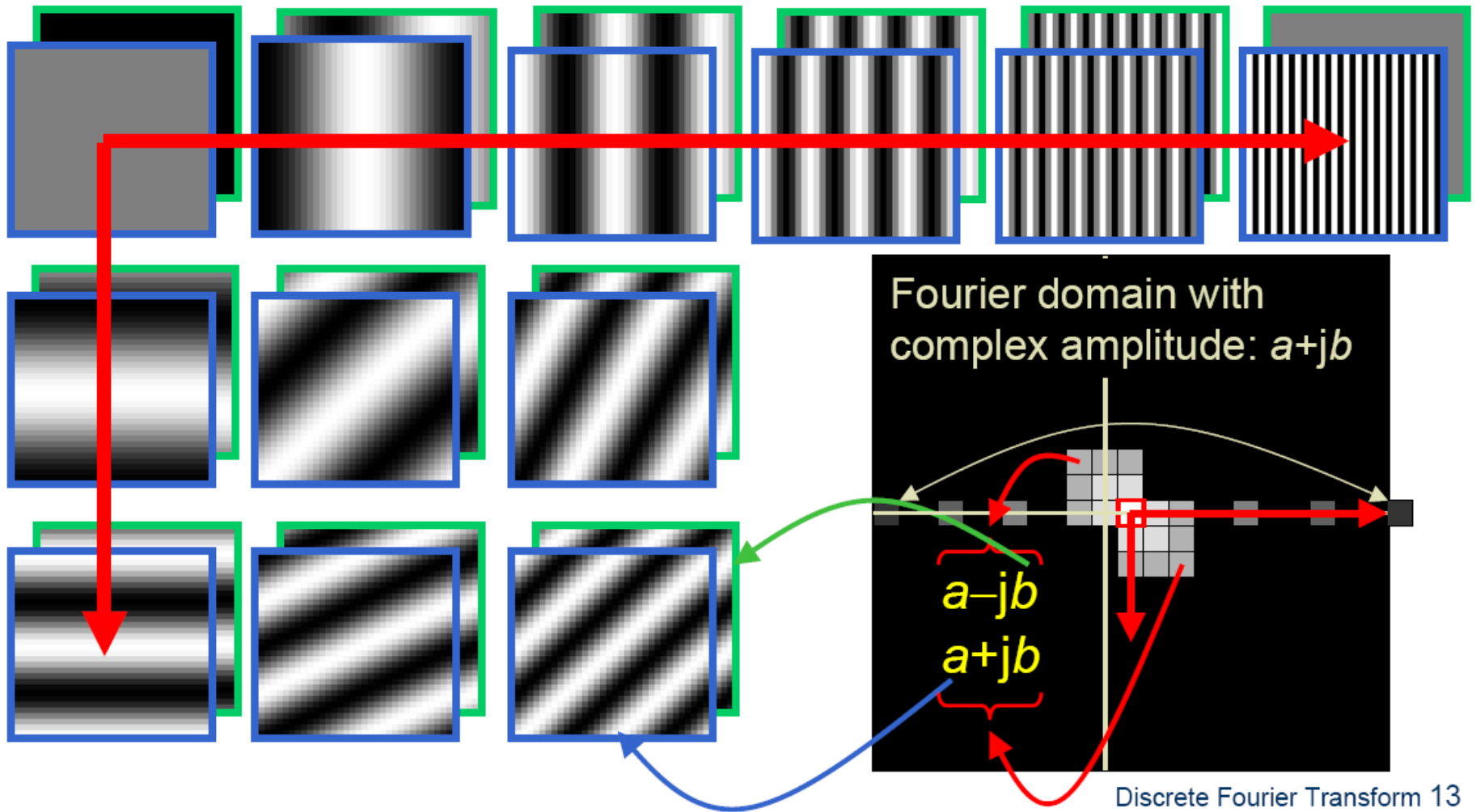
1	2	1
0	0	0
-1	-2	-1

Horizontal Edge  
(absolute value)



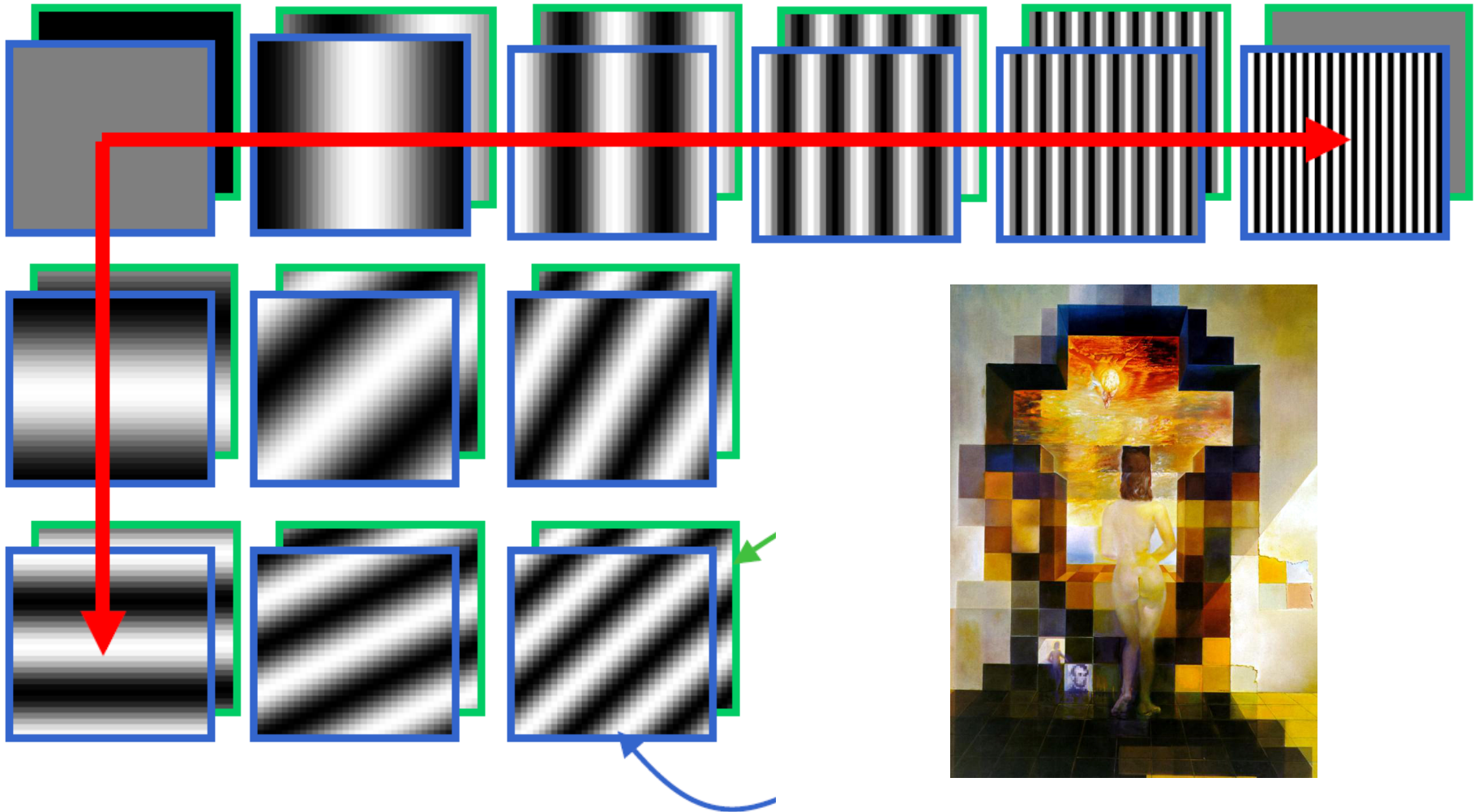
# Frequency View of Image

Thinking image in terms of frequencies at different magnitudes



# Frequency View of Image

Thinking image in terms of frequencies at different magnitudes





# Fourier Transform

## Recap

- Fourier transform stores the magnitude and phase at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of real and complex numbers

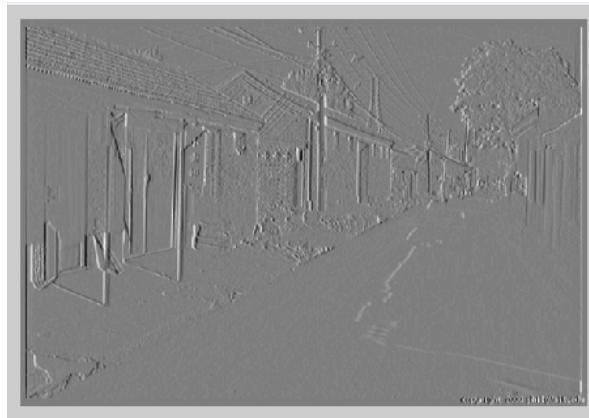
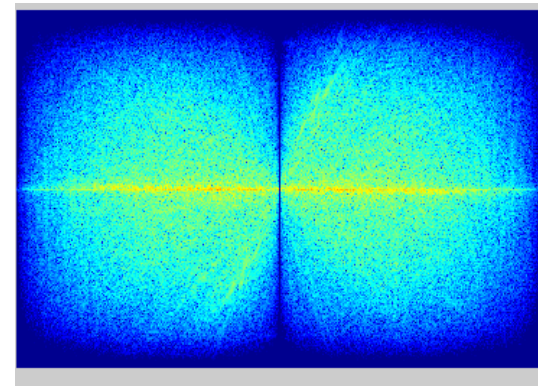
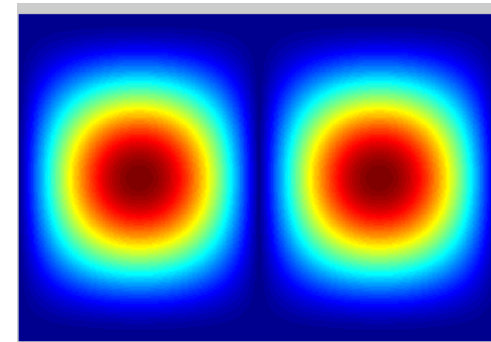
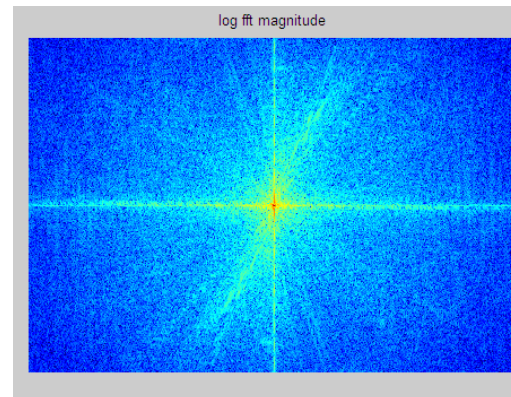
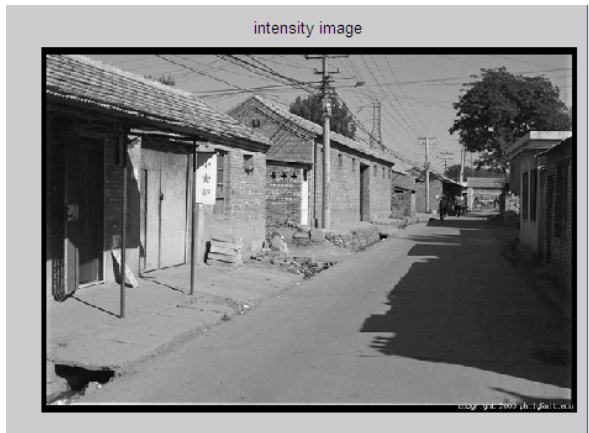

Continuous: 
$$H(\omega) = \int_{-\infty}^{\infty} h(x)e^{-j\omega x} dx$$

Discrete: 
$$H(k) = \frac{1}{N} \sum_{x=0}^{N-1} h(x)e^{-j\frac{2\pi kx}{N}}$$

# Filtering In Frequency Domain

Recap of Sobel kernel

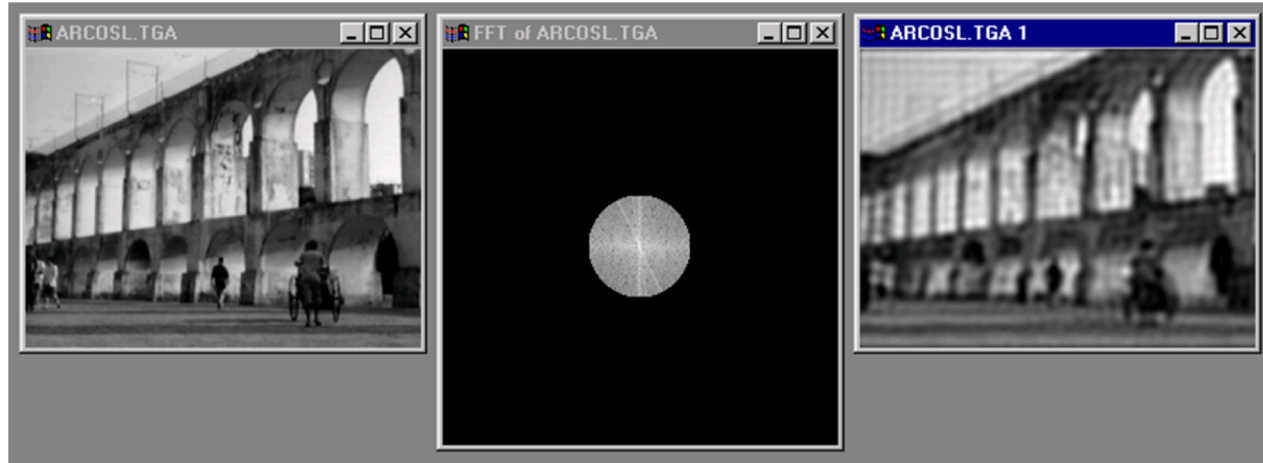
1	0	-1
2	0	-2
1	0	-1



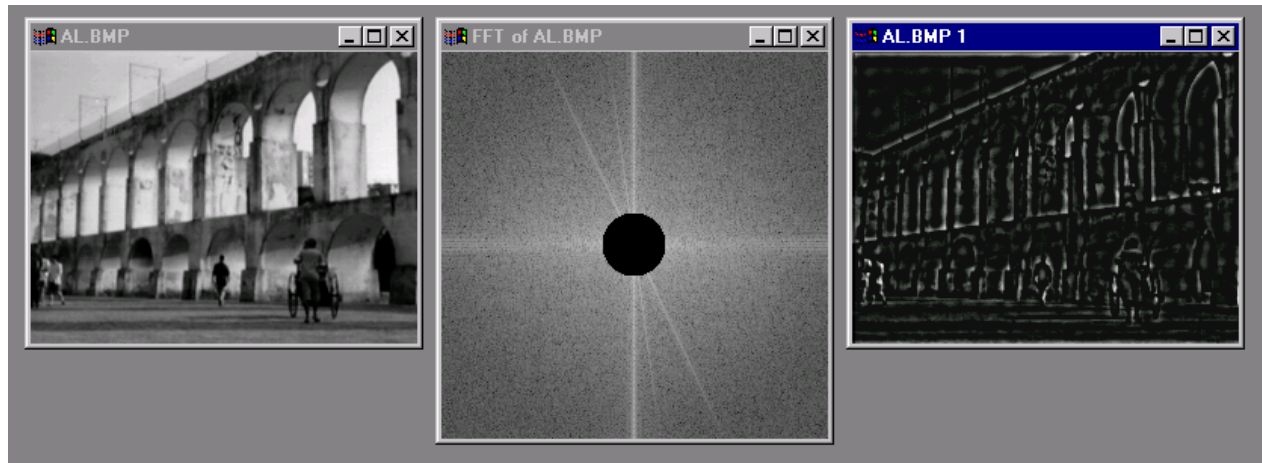
# Types Of Linear Filters

In terms of frequency

- Low-pass filter
  - Remove “high-frequency” components
  - Images become smoother

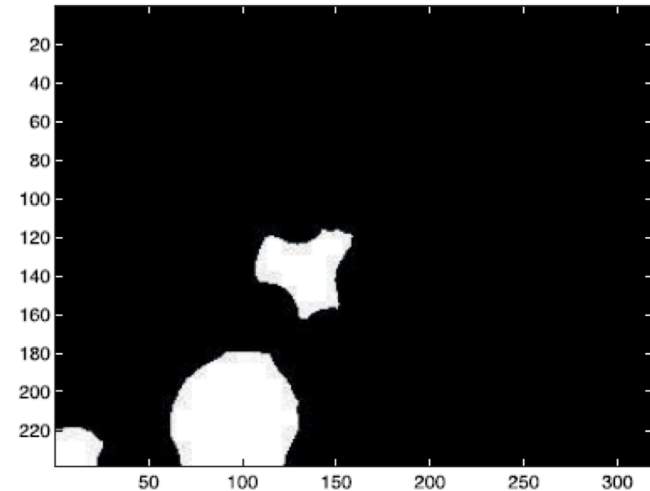
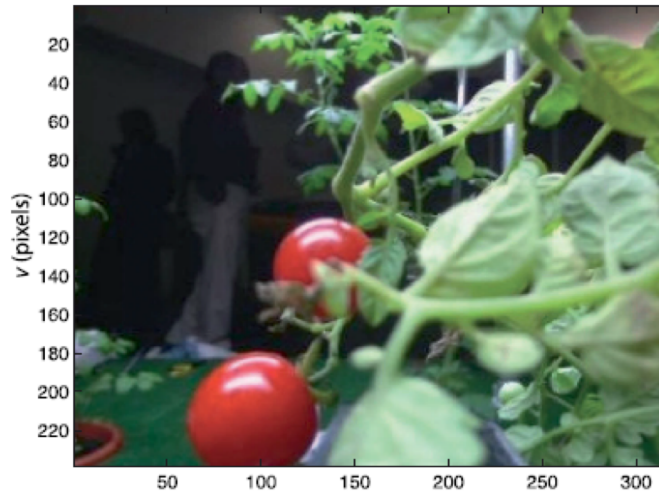
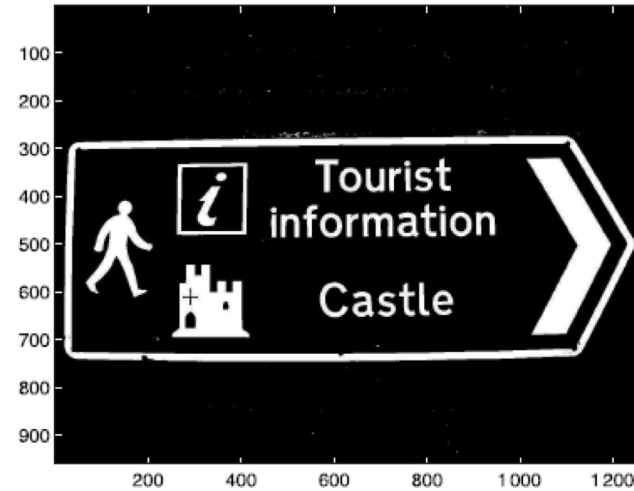
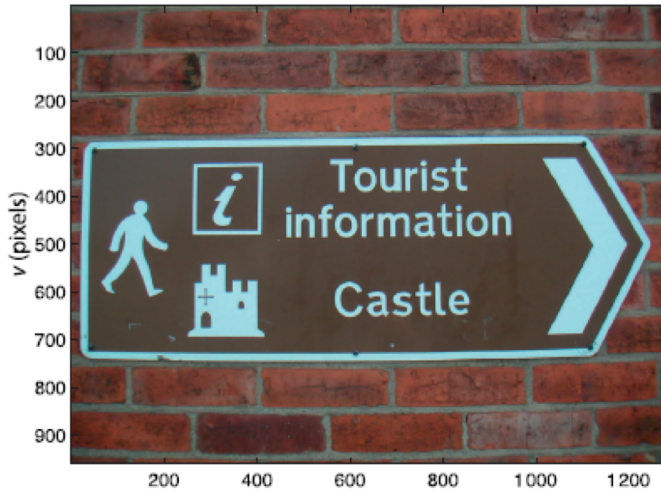


- High-pass filter
  - Remove “high-frequency” components
  - Image become sharper.



# Image Feature Extraction

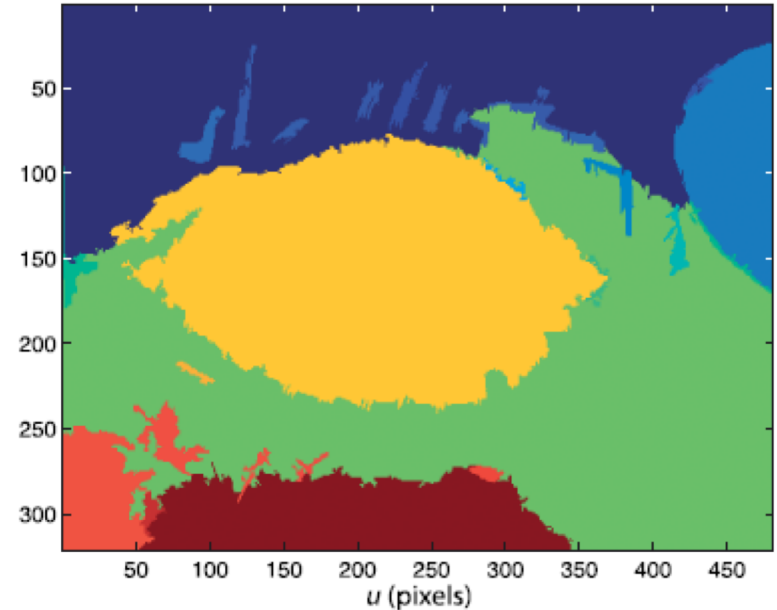
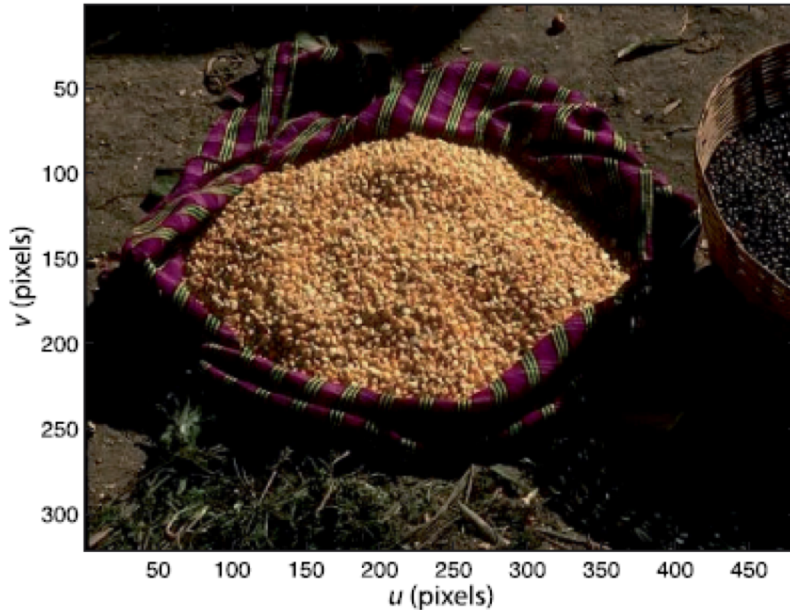
Motivation: information concentration





# Image Feature Extraction

Motivation: information concentration



- Reduces the data rate from 106–108 bytes/s at the output of a camera to order of tens of features per frame that can be used as input to a robot's control system.
- Features: Regions, Lines, Interest points

# Region Features

## Image segmentation

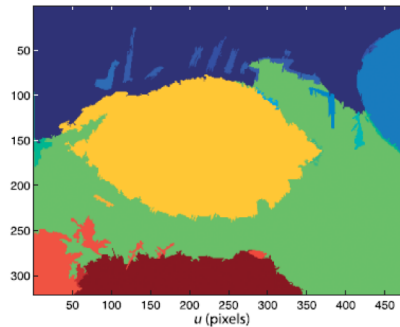
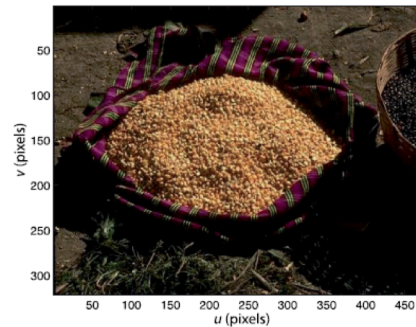


Each pixel is assigned to one of the  $C$  classes.  
Binary classification when  $C=2$ :

$$c[u, v] = \begin{cases} 0 & I[u, v] < t \\ 1 & I[u, v] \geq t \end{cases} \quad \forall (u, v) \in I$$

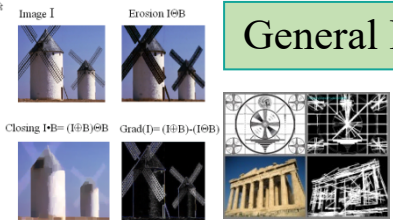
Adjacent pixels of the same class are *connected* to form spatial sets  $S_1 \dots S_m$

The sets  $S_i$  are *described* in terms of scalar or vector-valued *features* such as size, position, and shape.

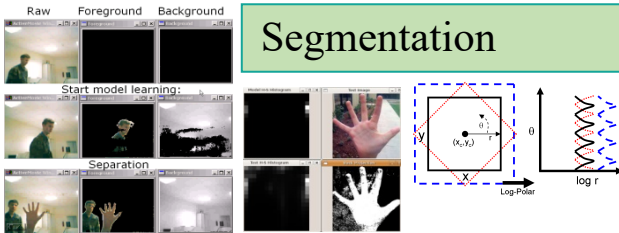


# OpenCV

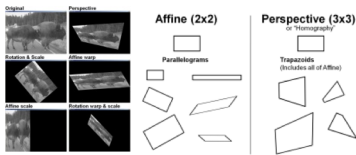
## General Image Processing Functions



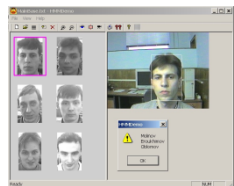
## Segmentation



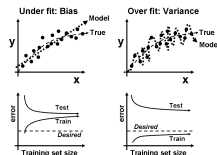
## Transforms



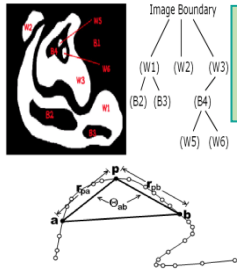
## Machine Learning: Detection, Recognition



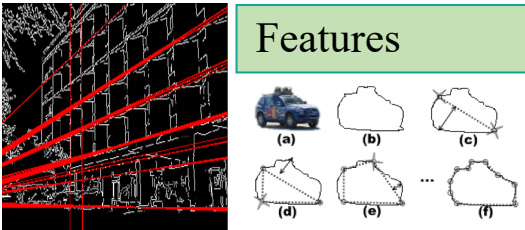
AncoraSIR.com



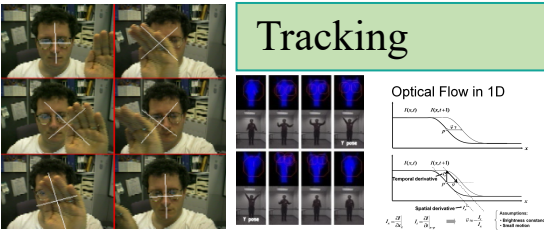
## Geometric Descriptors



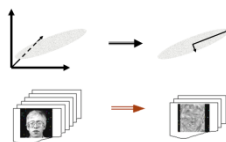
## Features



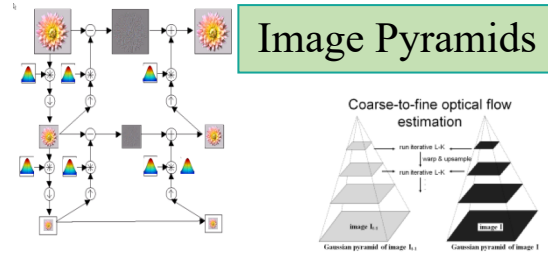
## Tracking



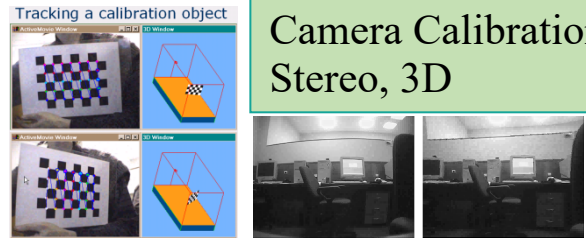
## Matrix Math



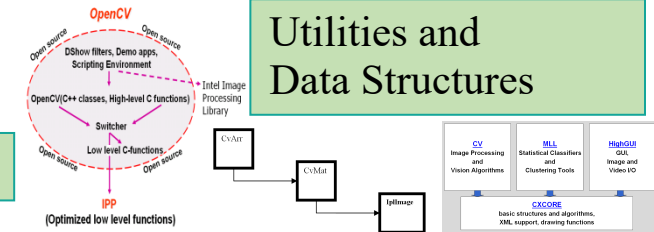
## Image Pyramids



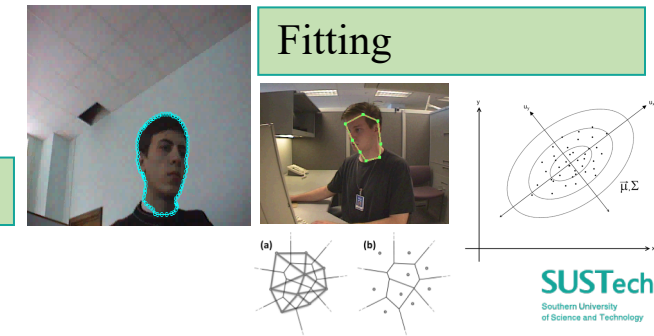
## Camera Calibration, Stereo, 3D



## Utilities and Data Structures



## Fitting



# OpenCV

---

## OpenCV-Python

- We are going to use OpenCV-Python API for our projects.
  - A Python wrapper around original OpenCV's C++ implementation.
- An appropriate tool for fast prototyping of computer vision problems
  - OpenCV combining with NumPy, SciPy, Matplotlib
- Reference
  - [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_tutorials.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html)



# OpenCV

---

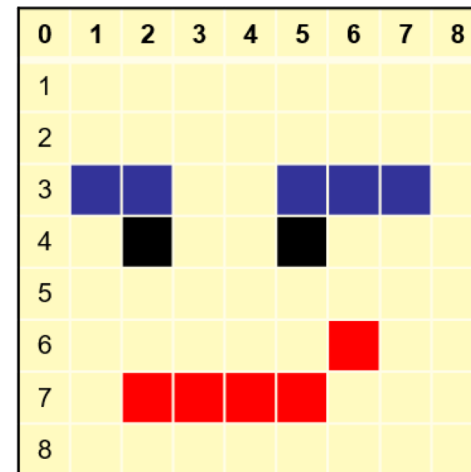
## Highlight

- Basic Operations on Images
- Image Thresholding
- Canny Edge Detection
- Contours in OpenCV
- Template Matching
- Image Segmentation with Watershed Algorithm

# Basic Operations on Images

- Data types
  - All the OpenCV array structures are converted to-and-from Numpy arrays.
- Reading/Display image
  - Color images returns an array of Blue, Green, Red values.

```
>>> import cv2
>>> import numpy as np
>>> img = cv2.imread('messi5.jpg')
>>> cv2.imshow(img)
# accessing only blue pixel
>>> blue = img[100,100,0]
>>> print blue
157
# modifying pixel value
>>> img[100,100] = [255,255,255]
```



# Basic Operations on Images

- Accessing image properties:
- Image ROI (Region of Interest)
  - Splitting/Merging Image Channels

```
>>> print img.shape  
(342, 548, 3)  
>>> print img.dtype  
uint8  
>>> ball = img[280:340, 330:390]  
>>> b,g,r = cv2.split(img)  
>>> img = cv2.merge((b,g,r))
```



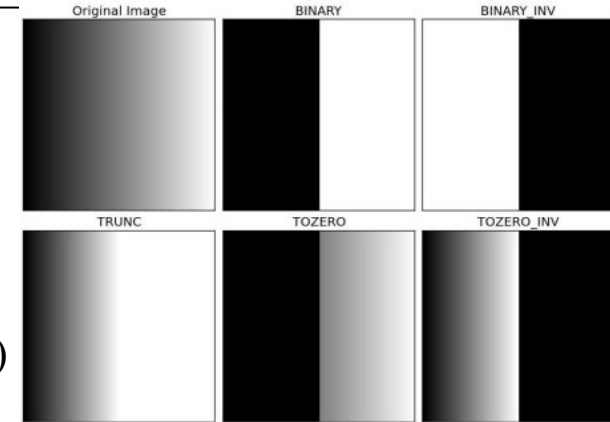
# Image Thresholding

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('gradient.png',0)
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)
titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

for i in xrange(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([],plt.yticks([]))

plt.show()
```



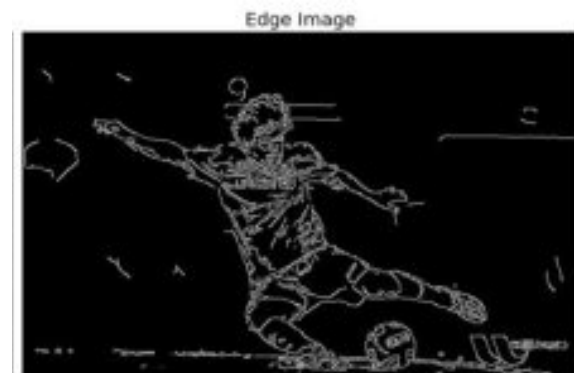
# Canny Edge Detection

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('messi5.jpg',0)
edges = cv2.Canny(img,100,200)

plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()
```



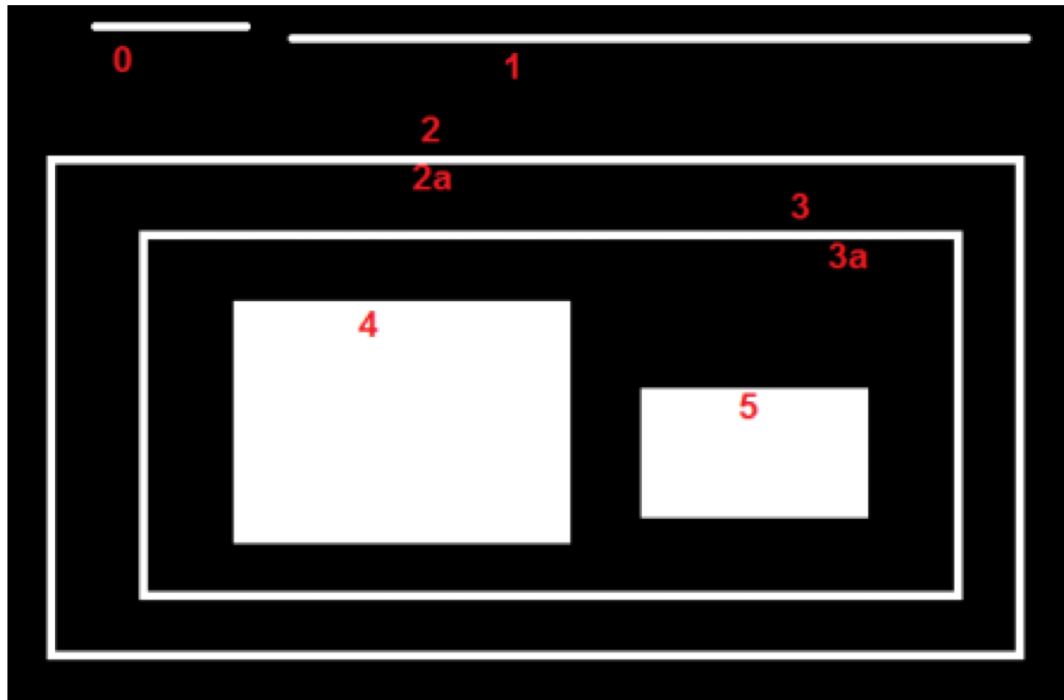
# Contours in OpenCV

- Contours can be explained simply as
  - a curve joining all the continuous points (along the boundary), having same color or intensity.
- Useful tool for shape analysis and object detection and recognition.
  - For better accuracy, use binary images.
  - So before finding contours, apply threshold or canny edge detection.

```
im = cv2.imread('test.jpg')
imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(imgray,127,255,0)
image, contours, hierarchy =
cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

# Contours in OpenCV

- Contour Hierarchy:
  - how one contour is connected to each other,
  - outer one as parent and inner one as child
- Contour Retrieval Mode:
  - RETR\_LIST, RETR\_TREE



# Contour Approximation for Shape Detection

- Contour approximation is predicated on the assumption that
  - a curve can be approximated by a series of short line segments.
- Contour approximation is implemented in OpenCV via
  - `theCv2.approxPolyDP`

```
# if the shape is a triangle, it will have 3 vertices
if len(approx) == 3:
    shape = "triangle"

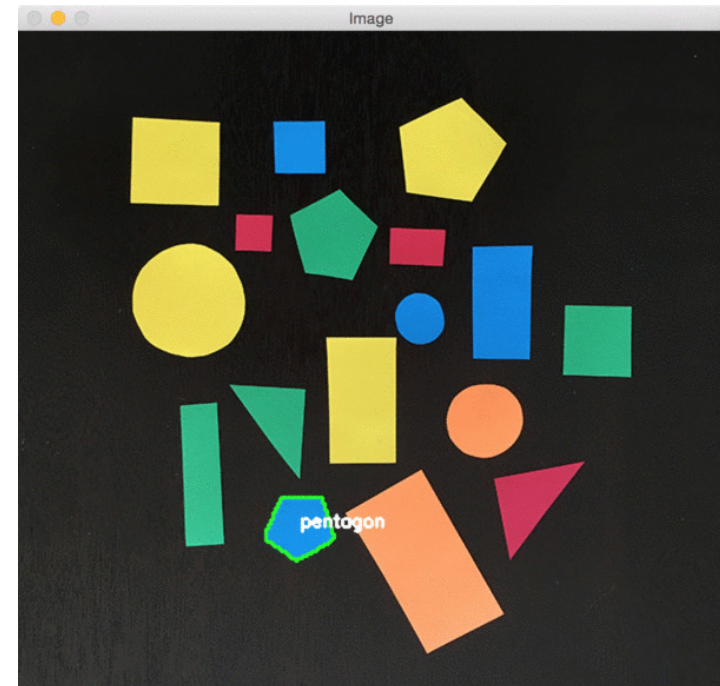
# if the shape has 4 vertices, it is either a square or
# a rectangle
elif len(approx) == 4:
    # compute the bounding box of the contour and use the
    # bounding box to compute the aspect ratio
    (x, y, w, h) = cv2.boundingRect(approx)
    ar = w / float(h)

    # a square will have an aspect ratio that is approximately
    # equal to one, otherwise, the shape is a rectangle
    shape = "square" if ar >= 0.95 and ar <= 1.05 else "rectangle"

# if the shape is a pentagon, it will have 5 vertices
elif len(approx) == 5:
    shape = "pentagon"

# otherwise, we assume the shape is a circle
else:
    shape = "circle"

# return the name of the shape
return shape
```





# Template Matching

---

- A method for searching and finding the location of a template image in a larger image.
  - slides the template image over the input image (as in 2D convolution)
  - compares the template and patch of input image under the template image.
- Several comparison methods are implemented in OpenCV

# Template Matching

Matching Result



Detected Point



Matching Result



Detected Point



# Template Matching

```
img = cv2.imread('messi5.jpg',0)
img2 = img.copy()
template = cv2.imread('template.jpg',0)
w, h = template.shape[::-1]

# All the 6 methods for comparison in a list
methods = ['cv2.TM_CCOEFF', 'cv2.TM_CCOEFF_NORMED', 'cv2.TM_CCORR',
           'cv2.TM_CCORR_NORMED', 'cv2.TM_SQDIFF', 'cv2.TM_SQDIFF_NORMED']

for meth in methods:
    img = img2.copy()
    method = eval(meth)

    # Apply template Matching
    res = cv2.matchTemplate(img,template,method)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
```

# Hough Circle Transform

- Hough Transform to find circles in an image



```
img = cv2.imread('opencv_logo.png',0)
img = cv2.medianBlur(img,5)
cimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)

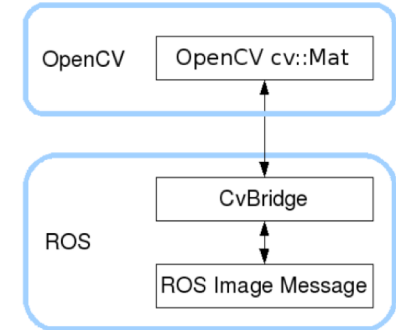
circles = cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,1,20,
                           param1=50,param2=30,minRadius=0,maxRadius=0)

circles = np.uint16(np.around(circles))
for i in circles[0,:]:
    # draw the outer circle
    cv2.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)
    # draw the center of the circle
    cv2.circle(cimg,(i[0],i[1]),2,(0,0,255),3)

cv2.imshow('detected circles',cimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

# ROS – OpenCV interface

- The OpenCV library is interfaced to ROS via ROS stack called *vision\_opencv* consisting of
  - *cv\_bridge*: converting between the OpenCV image data type and the ROS image message.
  - *image\_geometry*: correct the geometry of the image using calibration parameters

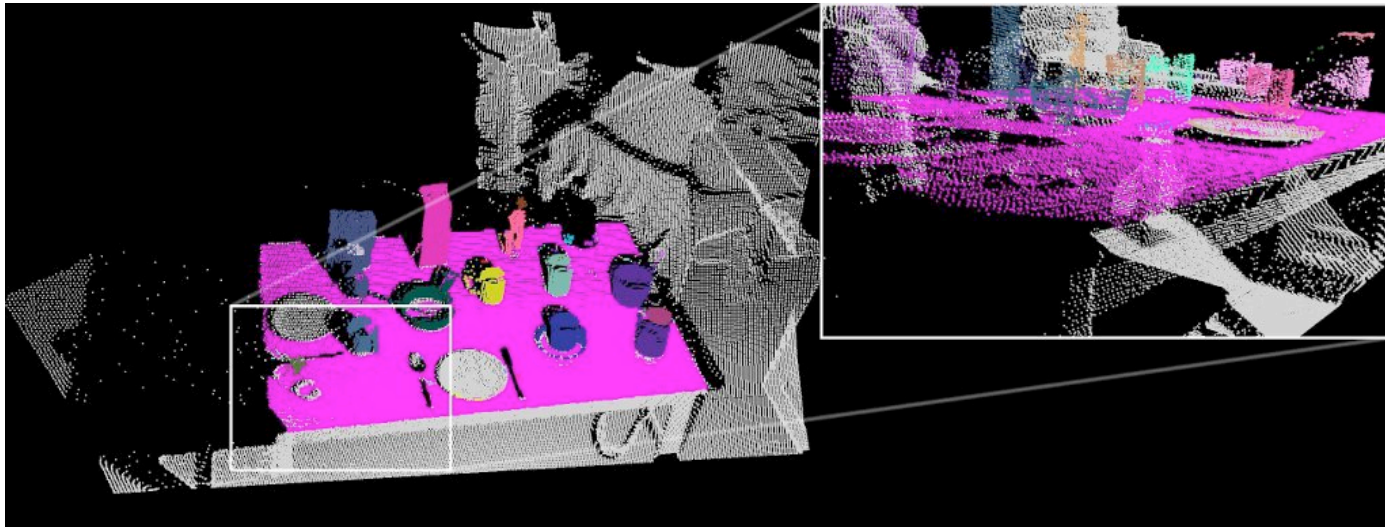


- Image processing using ROS and OpenCV
  - Subscribe the images from the camera driver from the topic `/usb_cam/image_raw (sensor_msgs/Image)`
  - Convert the ROS images to OpenCV image type using `CvBridge`
  - Process the OpenCV image using its APIs and find the edges on the image
  - Convert the OpenCV image type of edge detection to ROS image messages and publish into the topic `/edge_detector/processed_image`

# PCL

## Point Cloud Library

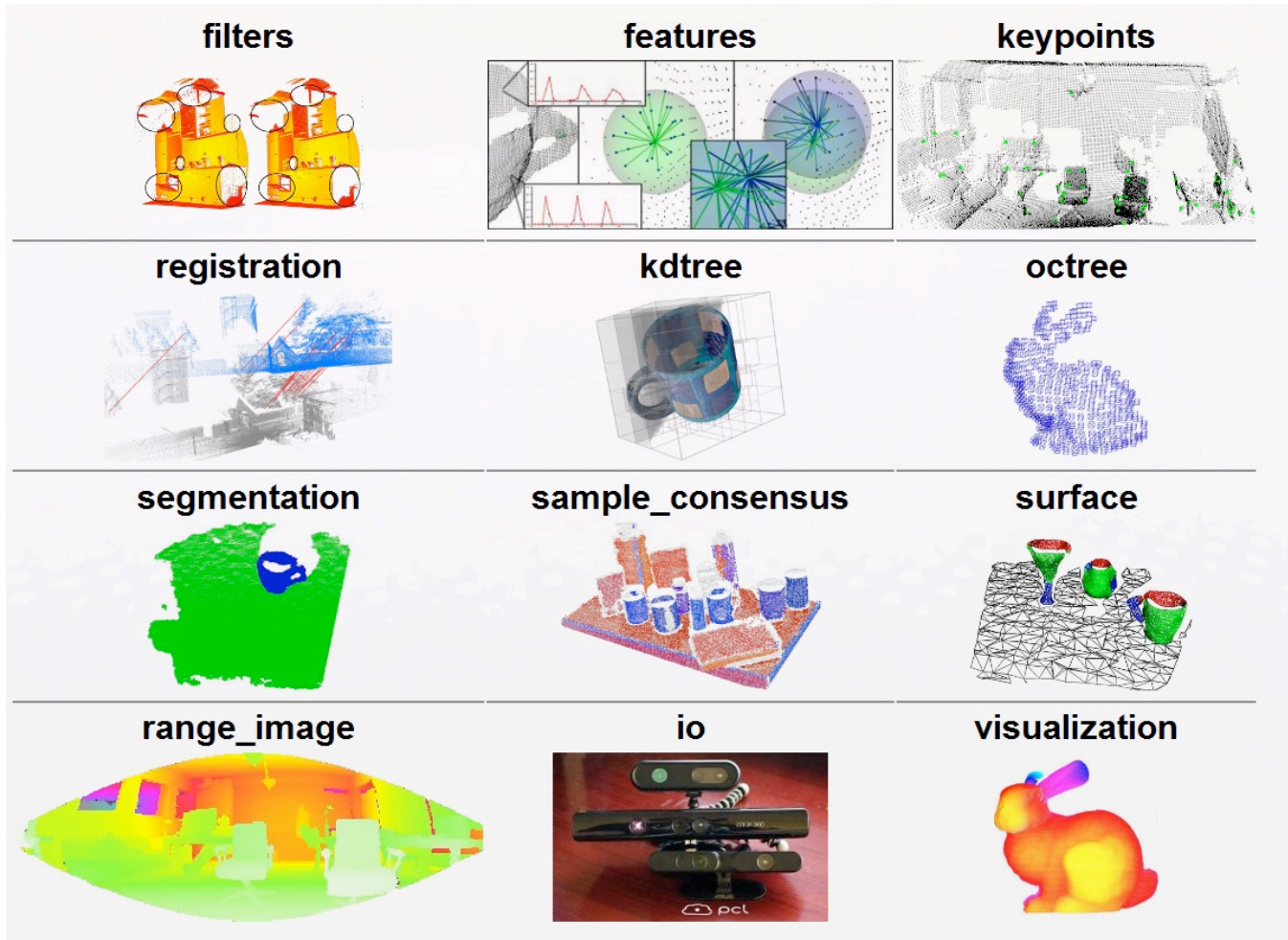
- Navigation / Obstacle avoidance
- Object recognition
- Grasping





# PCL

## Point Cloud Library Modules



# Homework

- Finish hand-eye calibration using robot hardware

ME336 Collaborative Robot Learning, Spring 2019

路径: 文件 > 参考资料



1987\_Least-Squares Fitting of two 3D points sets.pdf

WAN Fang 3月26日上传 · 版本1 · 565K

[在线预览](#) [上传新版本](#)

- Write a ROS node of image processing:
  - Recognize chess piece based on the circle feature
  - Locate the chess piece from RGB images
  - Compute the pick pose with reference to the robot coordinate
  - (Optional with **GREAT CARE**) Move the robot to the pick pose
- Codes:
  - <https://github.com/ancorasir/BionicDL-CobotLearning-Project>



# Thank you!

Prof. Song Chaoyang

- Dr. Wan Fang ([sophie.fwan@hotmail.com](mailto:sophie.fwan@hotmail.com))

