BSC~Microsoft Research Centre

Python for Machine Learning

Tassadaq Hussain

Professor Namal University

Director Centre for AI and Big Data

Collaborations:

Barcelona Supercomputing Center Barcelona, Spain

European Network on High Performance and Embedded Architecture and Compilation

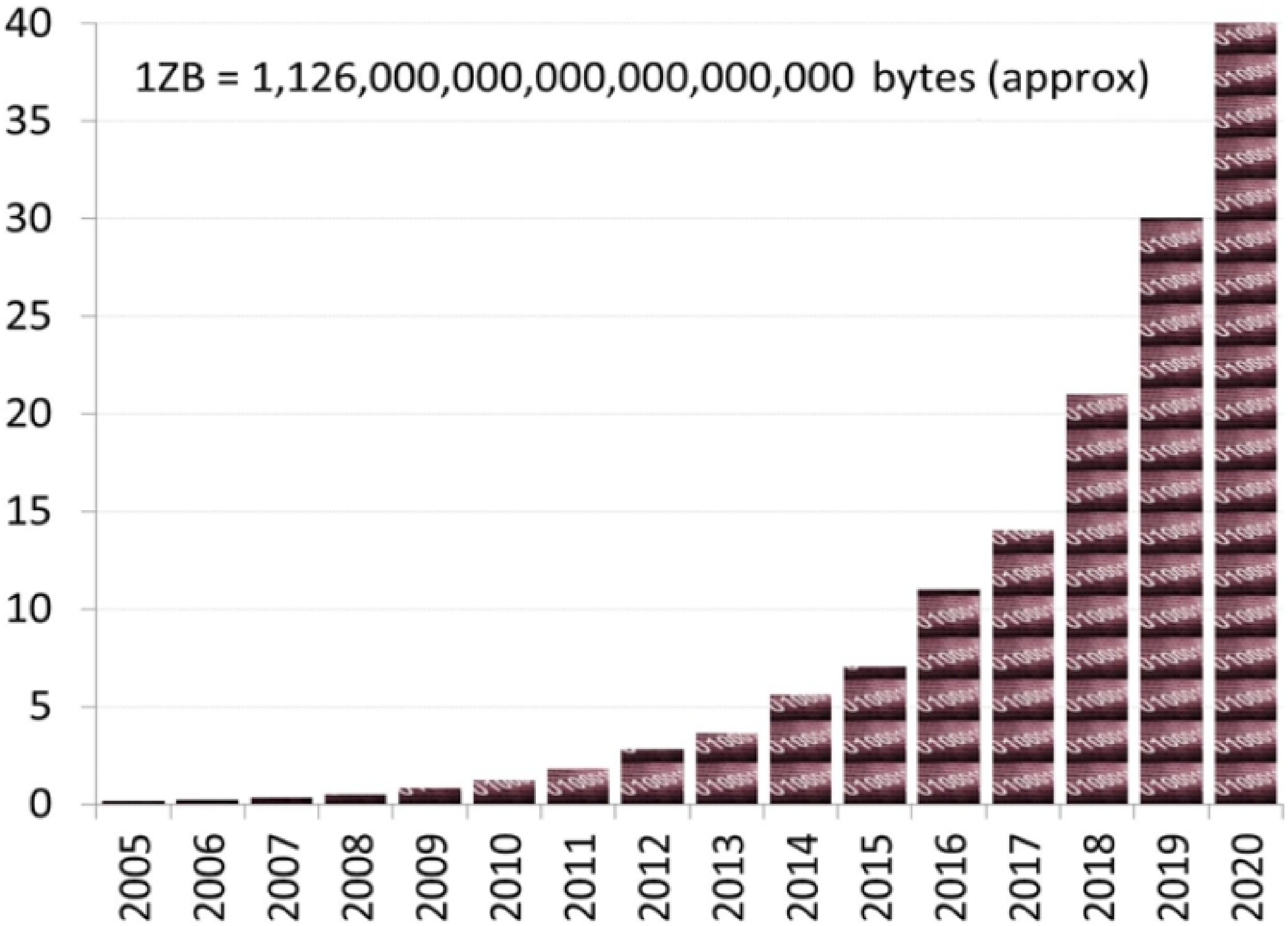
Pakistan Supercomputing Center

UCERD
Gathering
Intellectuals
www.ucerd.com

Agenda

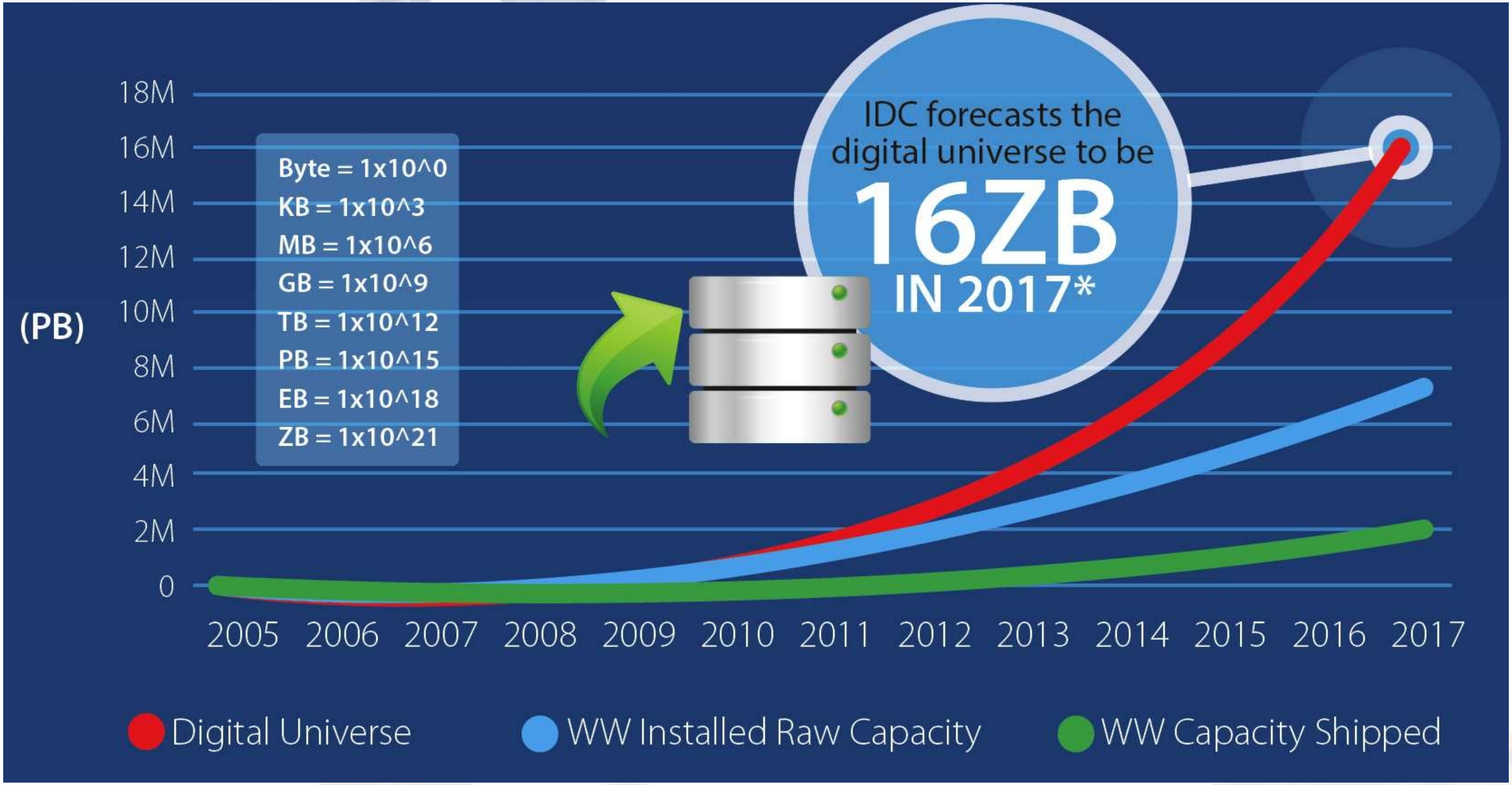
- **Data**
- Coding Languages
- Conventional Programming
- Python Language
- Python for Machine Learning
- Python for Image Processing

All Global Data in Zettabytes



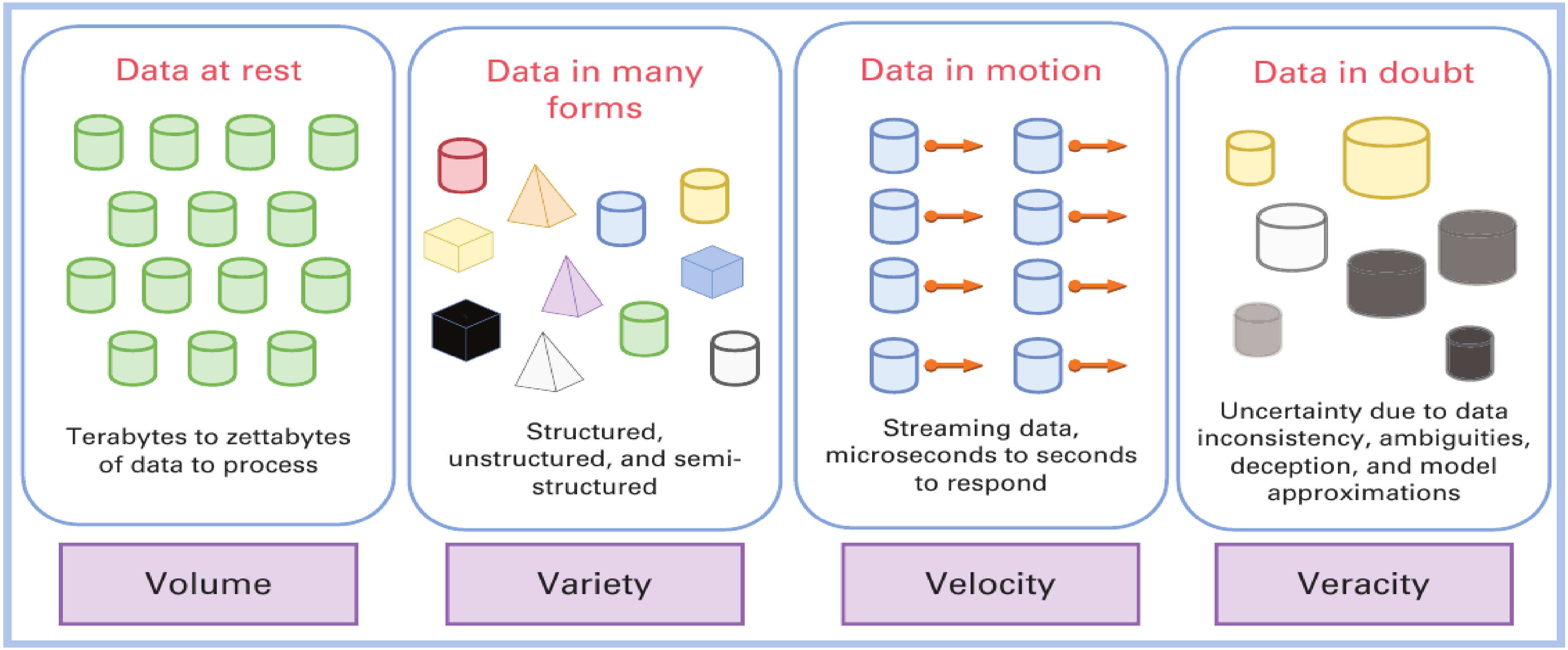
Source: http://www1.unece.org/stat/platform/display/msis/Big+Data

Information Future Trend



Technology Research Gartner Inc. states that information data volume doubles after every 18 months.

UCERD
Gathering Intellectuals
www.ucerd.com



Data at rest

Terabytes to zettabytes of data to process

Volume

Data in many forms

Structured, unstructured, and semi-structured

Variety

Data in motion

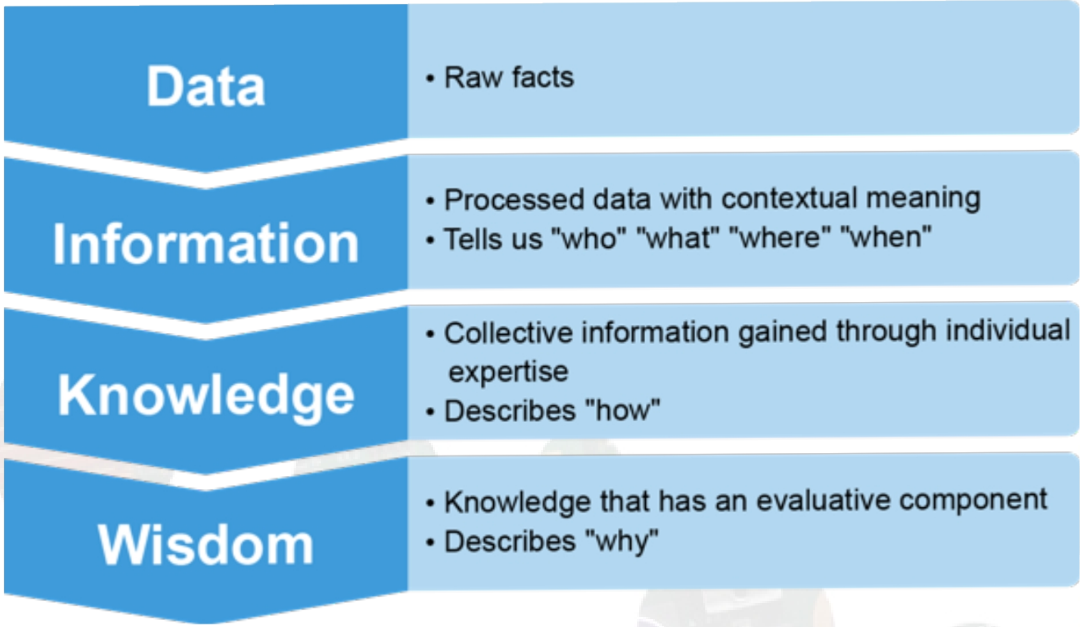
Streaming data, microseconds to seconds to respond

Velocity

Data in doubt

Uncertainty due to data inconsistency, ambiguities, deception, and model approximations

Veracity



Data

- Raw facts

Information

- Processed data with contextual meaning
- Tells us "who" "what" "where" "when"

Knowledge

- Collective information gained through individual expertise
- Describes "how"

Wisdom

- Knowledge that has an evaluative component
- Describes "why"

Processing Tech

Data Type:

- A data type represents the type or kind of data that a variable or object can hold in a programming language.
- It defines what operations can be performed on the data, how much memory it occupies, and how the data is stored.
- Examples of common data types include integers, floating-point numbers, strings, and booleans.

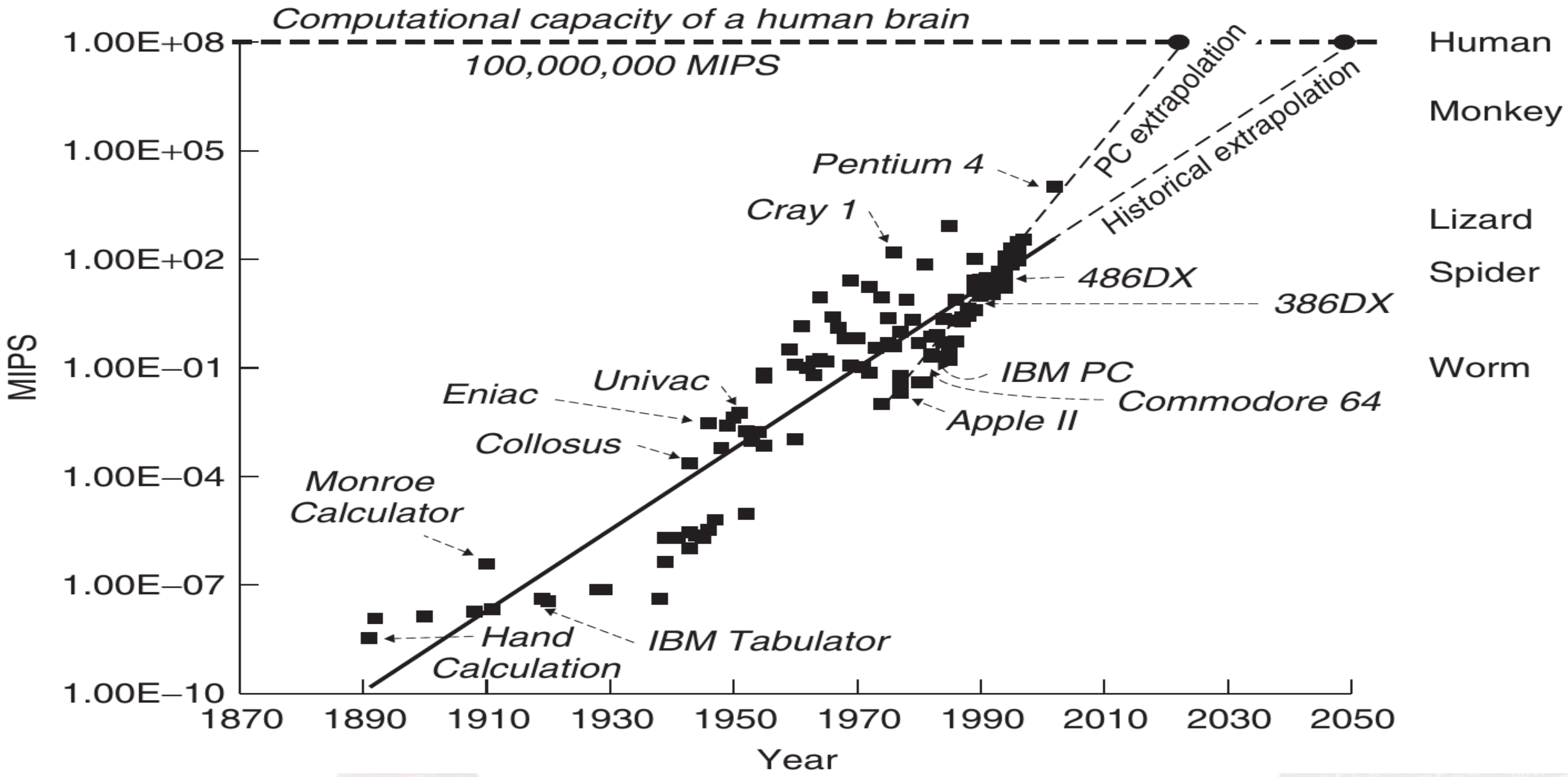
Data Set:

- A data set is a collection of data points or records that are related in some way.
- It often represents a sample or a population of data used for analysis, research, or study.
- Data sets can be organized in various structures like tables, spreadsheets, databases, or even files.

Data Structure:

- A data structure is a way of organizing and storing data in a computer's memory or storage devices.
- It defines how data elements are arranged, accessed, and manipulated.
- Data structures can be simple, like arrays and linked lists, or complex, like trees and graphs.

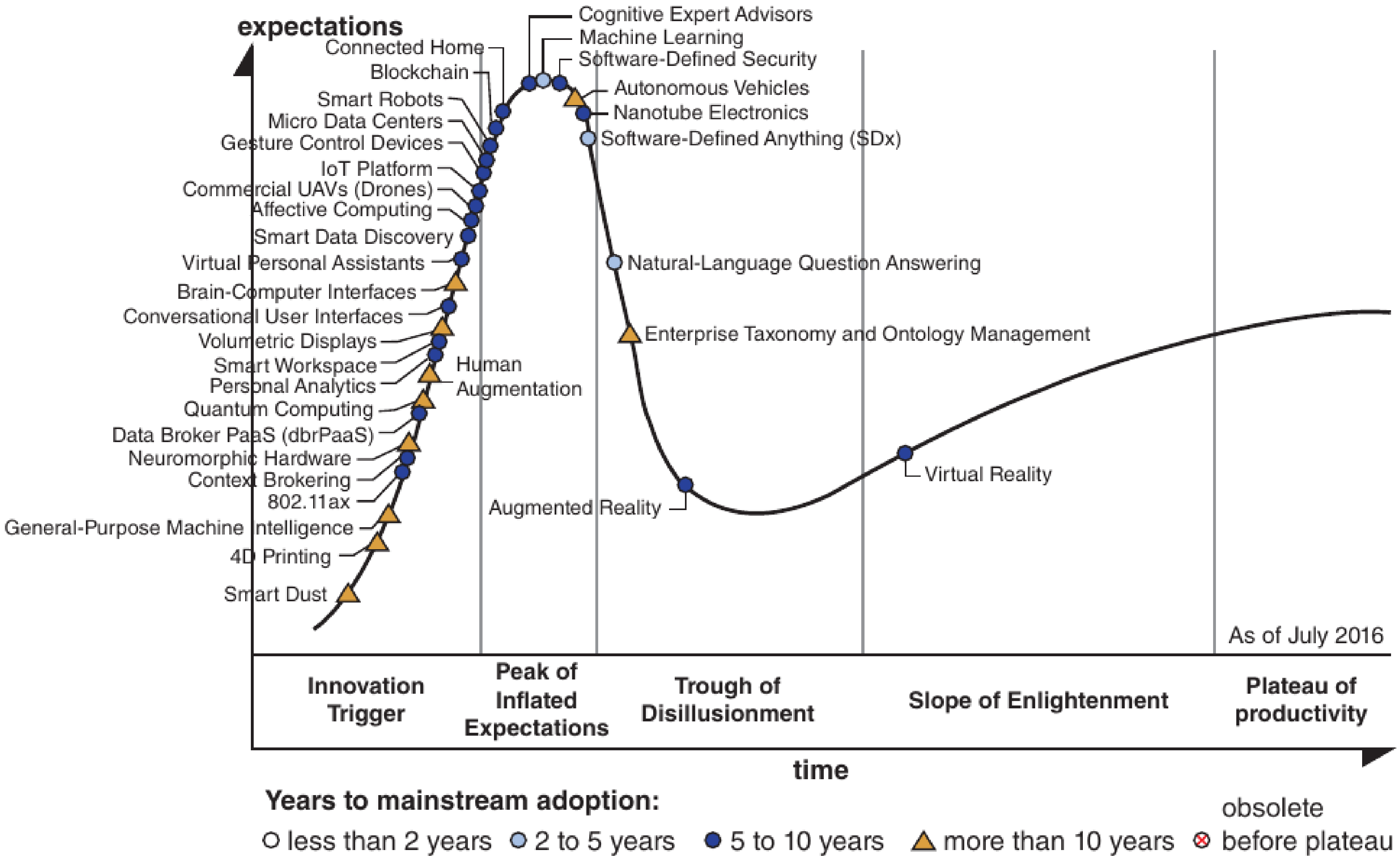
Computational Capability



It is estimated that sometime between the years **2025 and 2050**, a **personal computers** will exceed the calculation power of a human brain.

UCERD
Gathering Intellectuals
www.ucerd.com

Hype cycle for emerging high technologies to reach maturity and industrial productivity within the next decade.

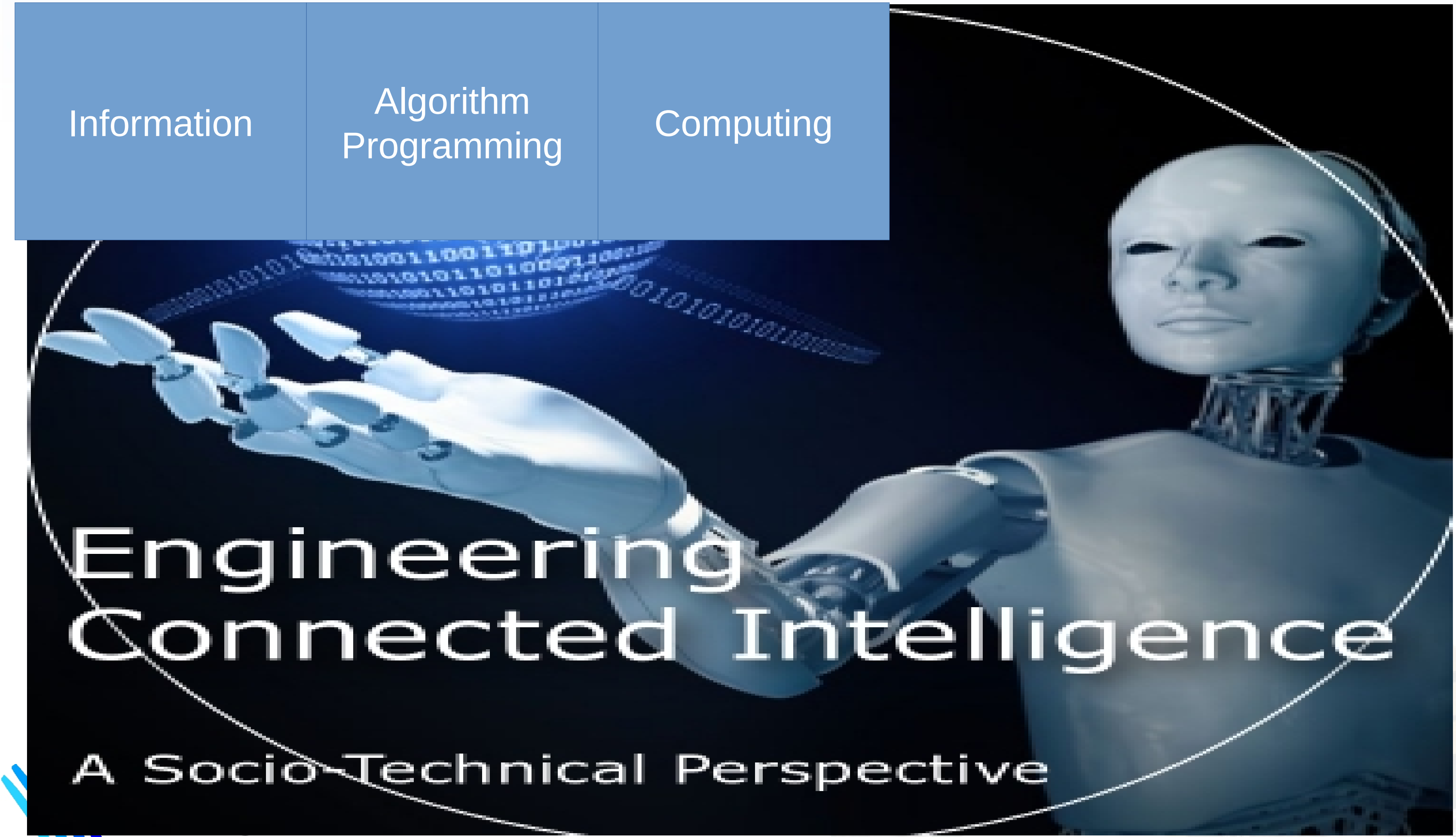


Intellectual Capability

Information

Algorithm Programming

Computing



Agenda

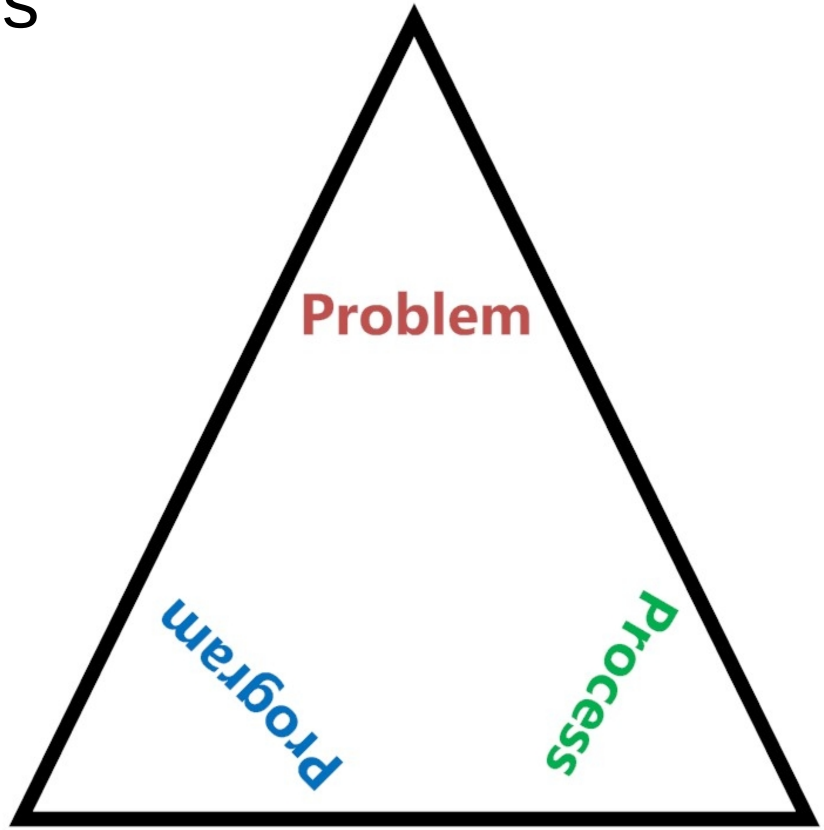
- Importance
- **Coding Languages**
- Conventional Programming
- Python Language
- Python for Machine Learning
- Python for Image Processing

UCERD
Gathering
Intellectuals
www.ucerd.com

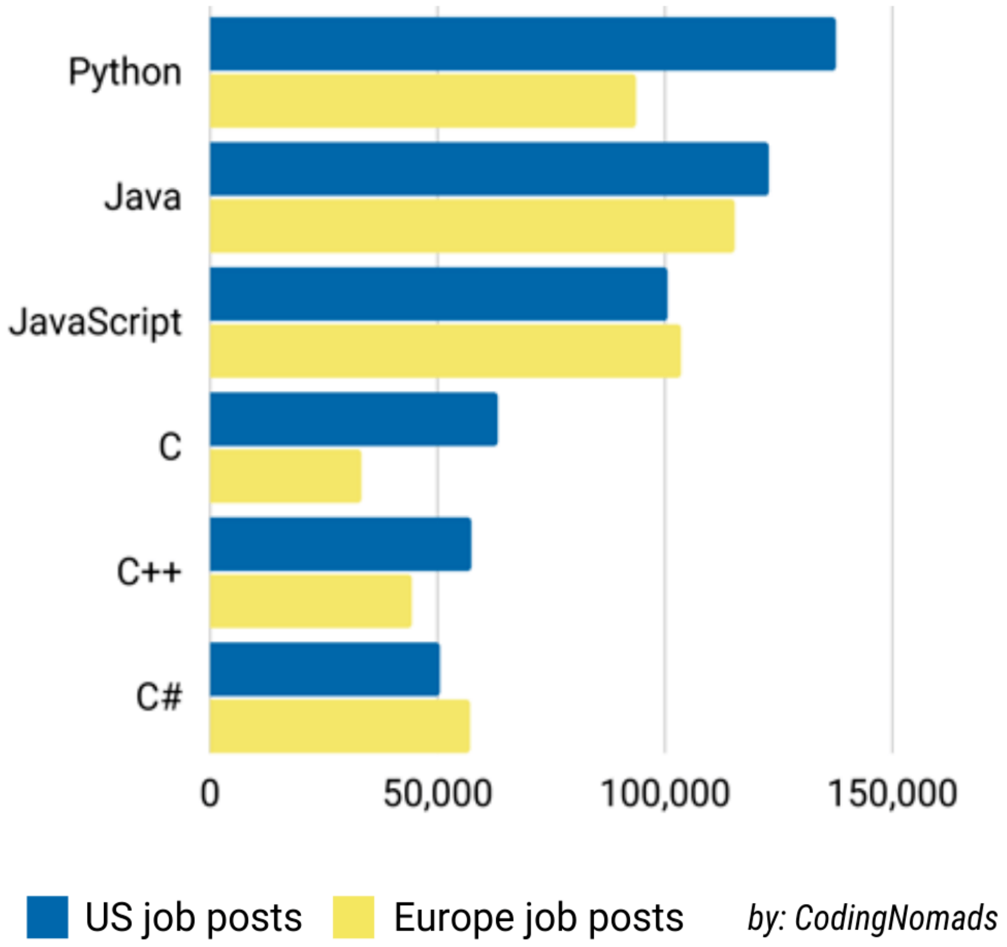
Data Processing

Coding Languages

- Programming Languages
- Scripting Languages



Most in-demand programming languages 2021-2022



Top Programming Languages: Rankings In Comparison

Index	1st	2nd	3rd	4th	5th
IEEE Spectrum	Python	Java	C	C++	JavaScript
GitHut 2.0 (Pull Requests)	JavaScript	Python	Java	Go	Ruby
GitHut 2.0 (Pushes)	JavaScript	Python	Java	C++	PHP
RedMonk	JavaScript	Python	Java	PHP	C++/C#
PYPL	Python	Java	Javascript	C#	C/C++
TIOBE	C	Python	Java	C++	C#

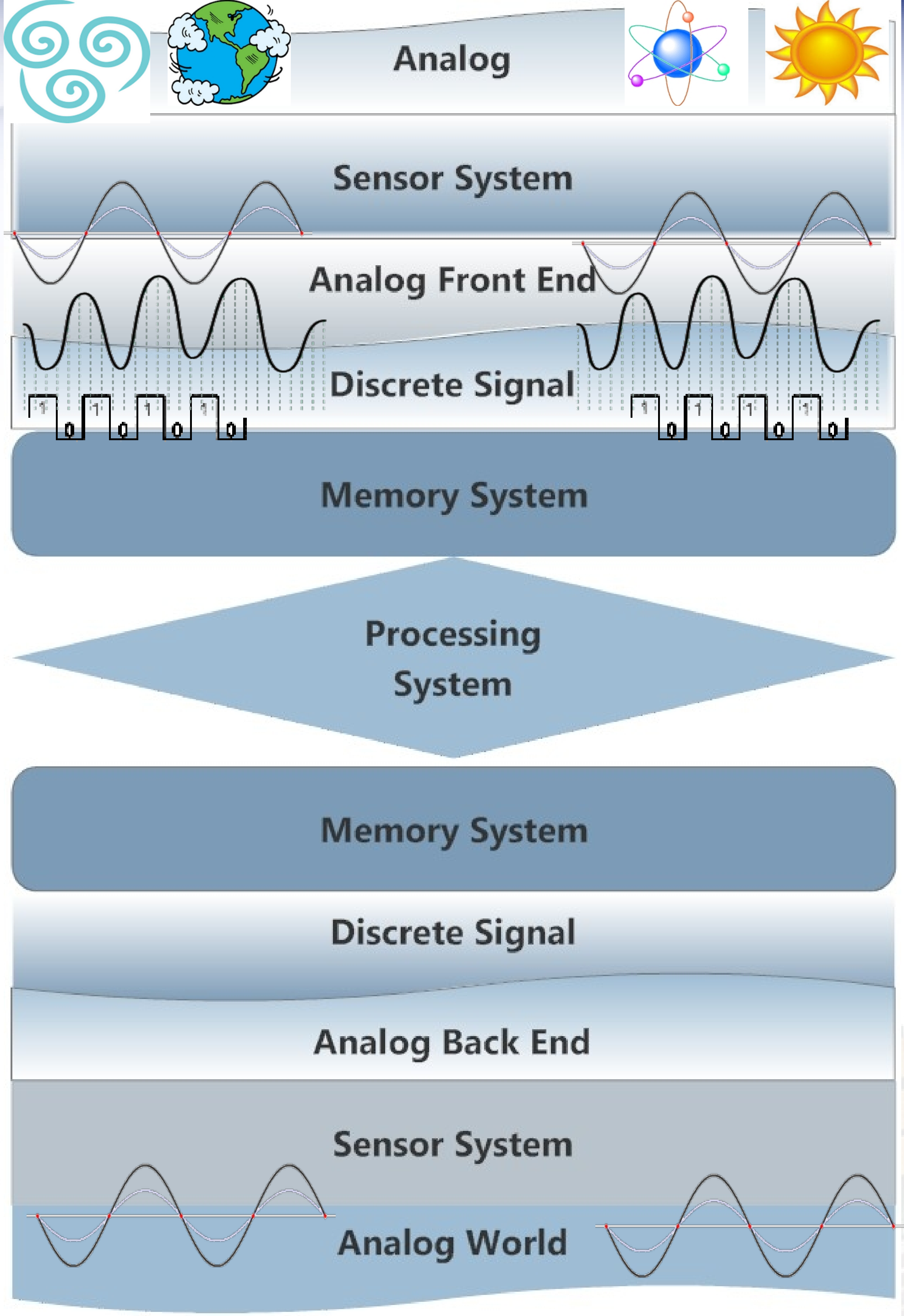
Agenda

- Importance
- Coding Languages
- **Conventional Programming**
- Python Language
- Python for Machine Learning
- Python for Image Processing

Signal Processing Applications

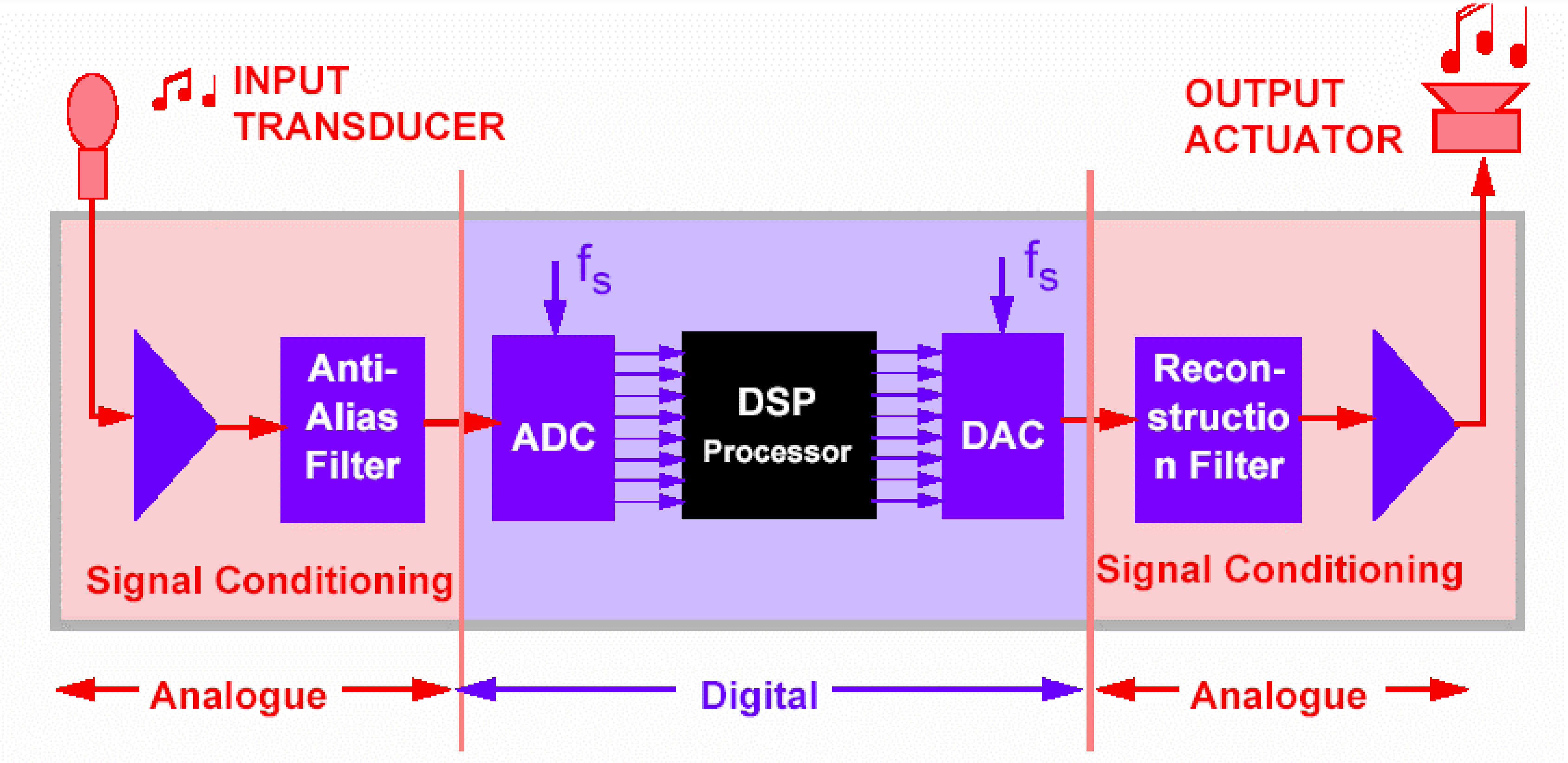
- Sound applications
  - Compression, enhancement, special effects, synthesis, recognition, echo cancellation,…
  - Cell Phones, MP3 Players, Movies, Dictation, Text-to-speech,…
- Communication
  - Modulation, coding, detection, equalization, echo cancellation,…
  - Cell Phones, dial-up modem, DSL modem, Satellite Receiver,…
- Automotive
  - ABS, GPS, Active Noise Cancellation, Cruise Control, Parking,…
- Medical
  - Magnetic Resonance, Tomography, Electrocardiogram,…
- Military
  - Radar, Sonar, Space photographs, remote sensing,…
- Image and Video Applications
  - DVD, JPEG, Movie special effects, video conferencing,…
- Mechanical
  - Motor control, process control, oil and mineral prospecting,…

Signal Processing System



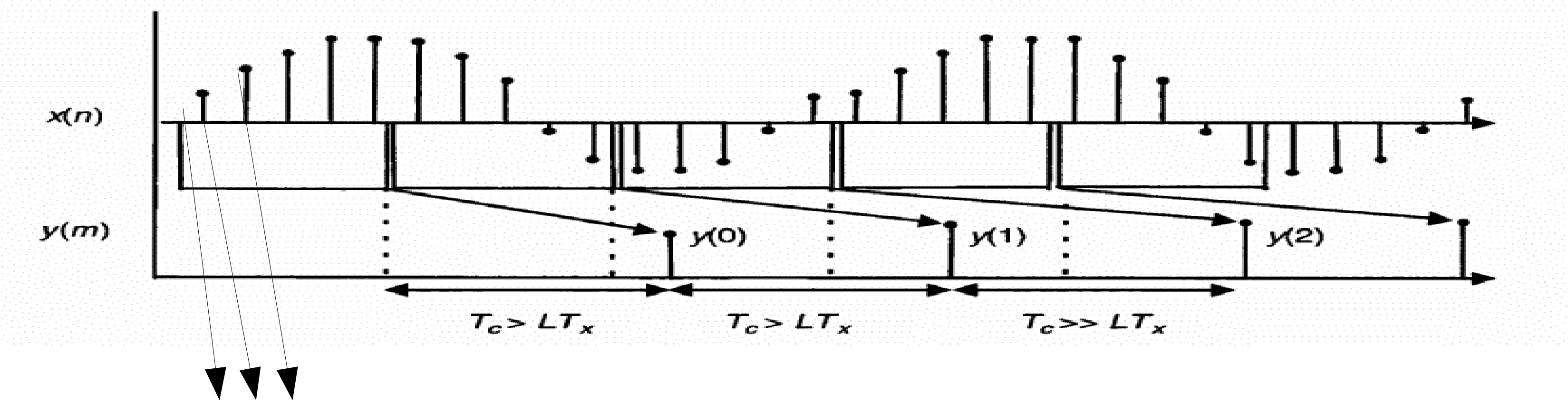
UCERD
Gathering
Intellectuals
www.ucerd.com

Signal Processing System



UCERD
Gathering Intellectuals
www.ucerd.com

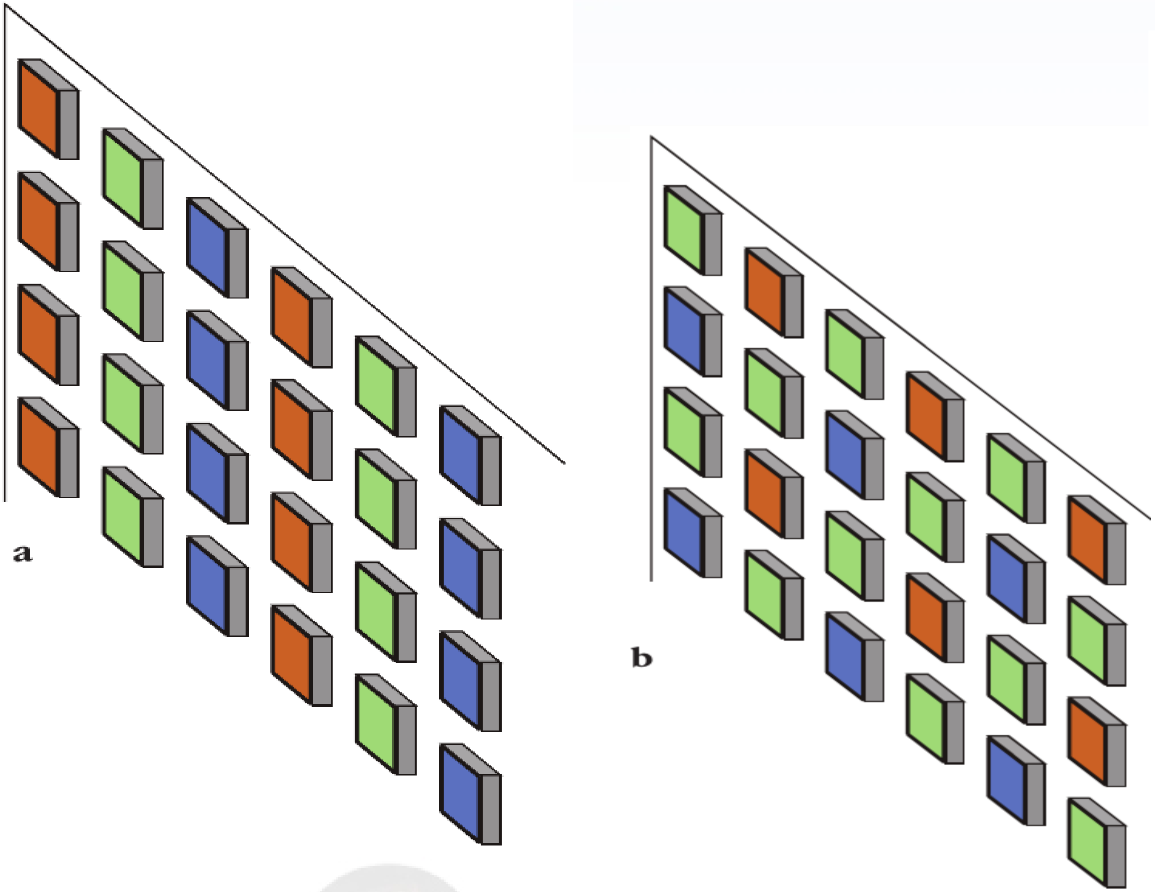
Signals

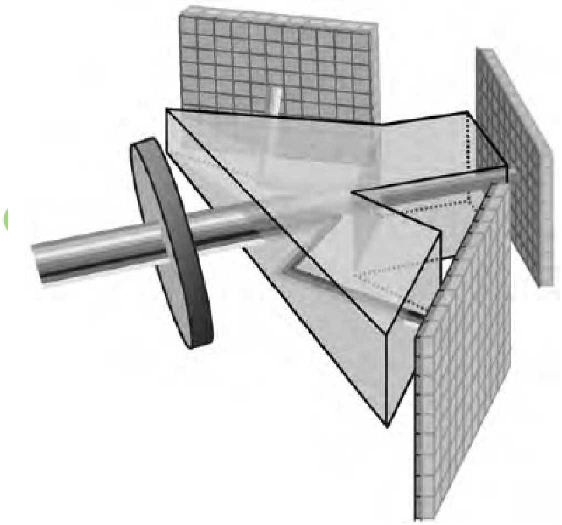


0,1,2,2.2,3,3,2.9,2,1,-.5,-2 - -------------------------

x[100]=0,1,2,2.2,3,3,2.9,2,1,-.5,-2]

Three-chip color Camera

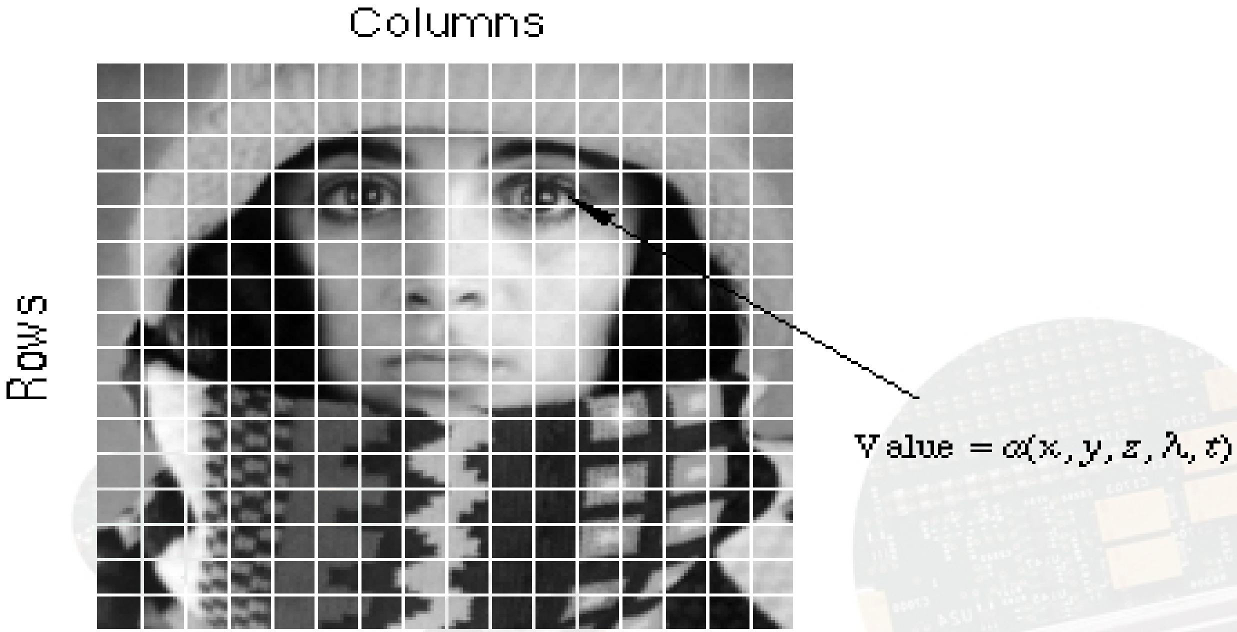




(a) Bayer (b) Filter patterns used in single chip cameras.

Color Pixel = **Red** (8bit) + **Green** (8bit) + **Blue** (8bit)

Gray scale intensity = 0.299 R + 0.587 G + 0.114 B



Pixel >> Image >> Video

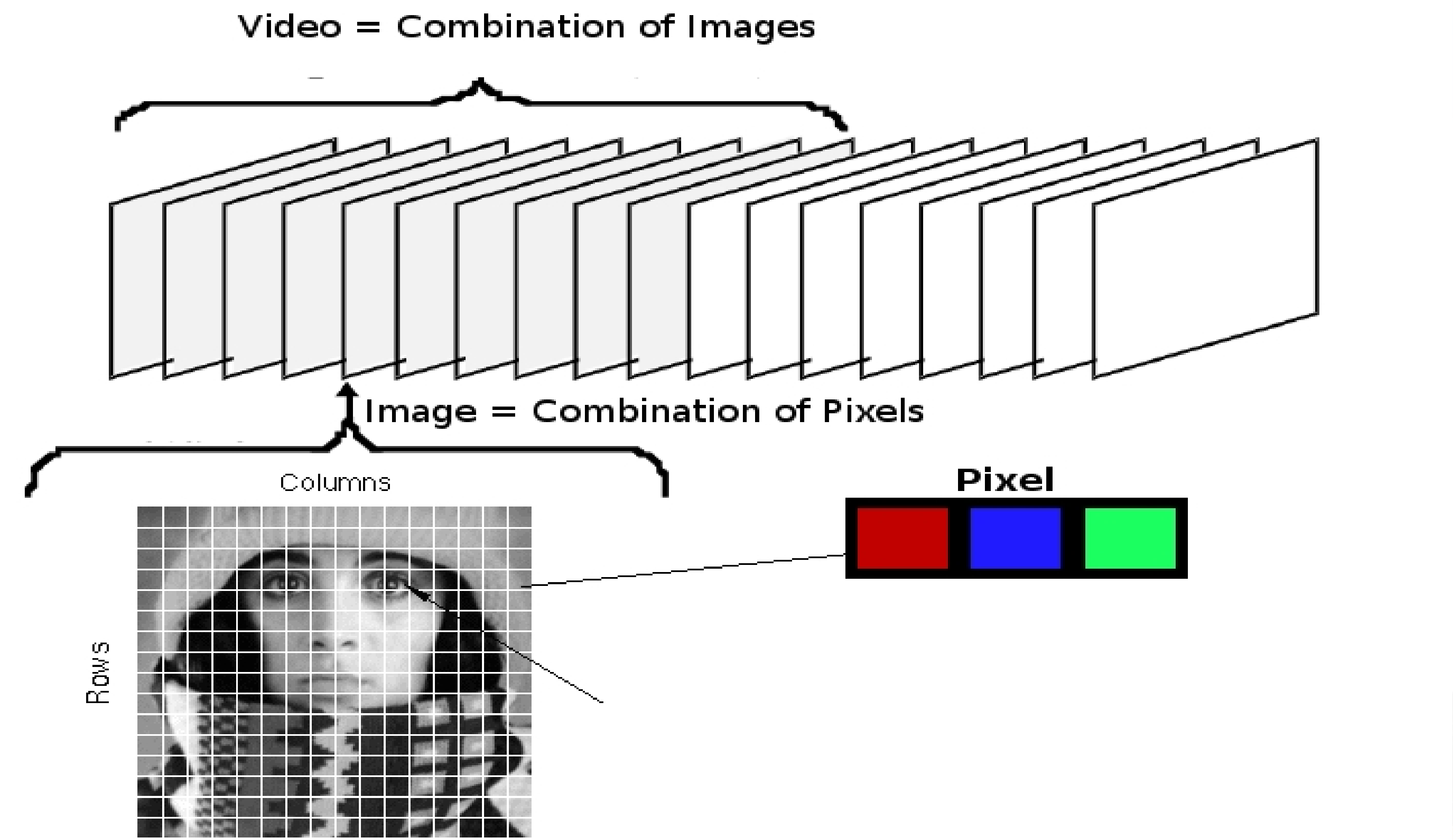
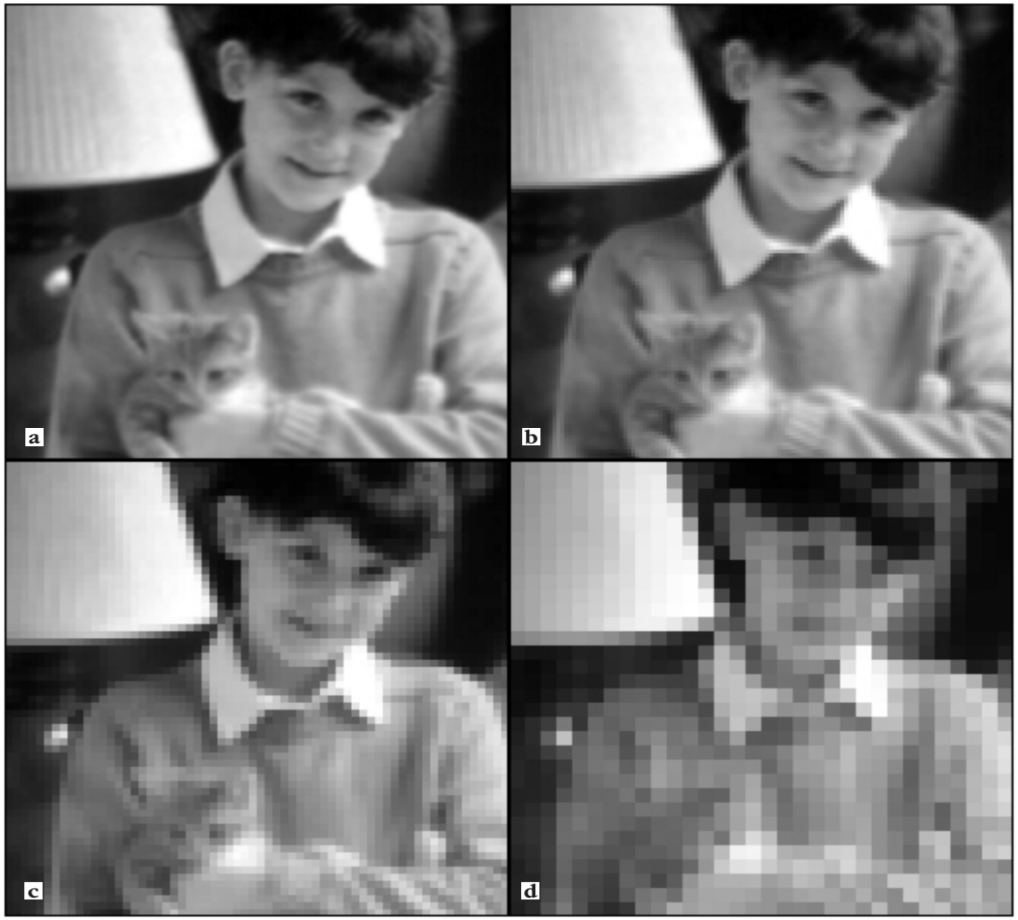


Image Resolution



(a) 256 × 256; (b) 128 × 128; (c) 64 × 64, (d)32 × 32.

Pixel Depth

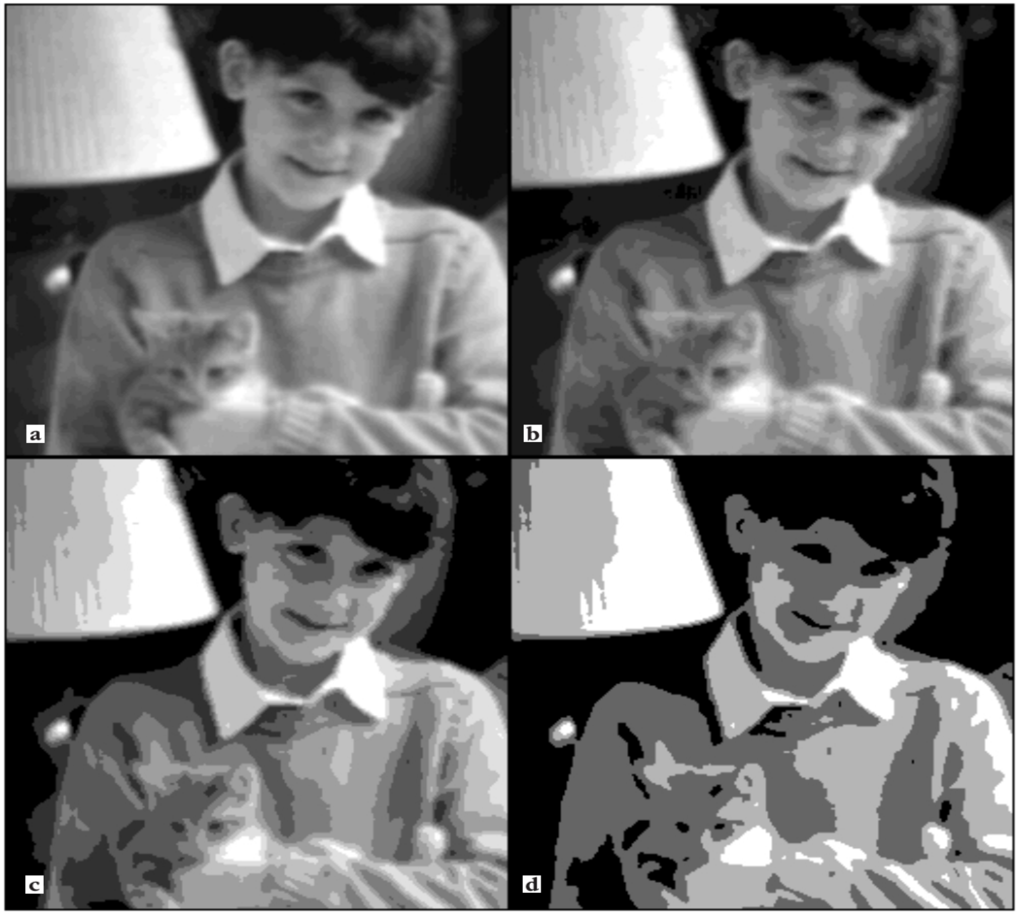


Image 256x256 array pixels: (a) 32 bit (b) 16 (c) 8 (d) 4

Performance Measures

- 3 Mega Pixel Image = 3145720 pixels
- A 32 bit Processor = 3.14 million operation / sec

  Pixels = 2048 x 1536 x 24 bits/pixel
- Local Memory = 9.4 Mega Byte for single Image
- Video Processing = $3.14 \times 10^6 \times 30$ (fps)

  $= 94.2 \times 10^6$

Levels of processing

Scalar Processing

- Perform single operation on a single signal value

Stream Processing

- All computations with one input sample are completed before the next input sample arrives

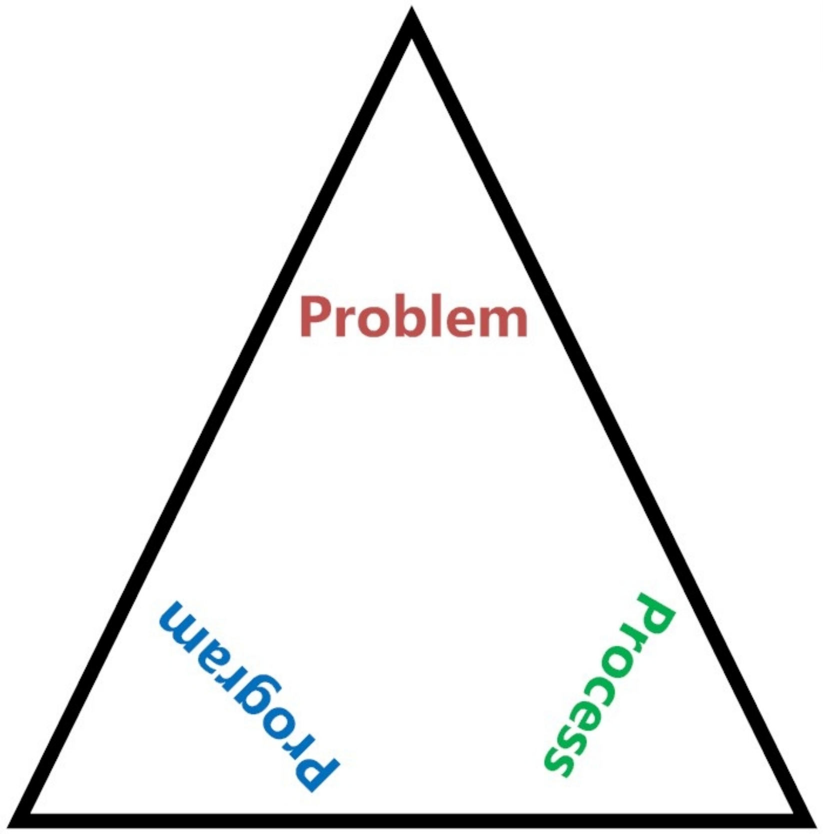
Block processing

- Each input sample x(n) is stored in memory before any processing occurs upon it. After *L* input samples have arrived, the entire collection of samples is processed at once.

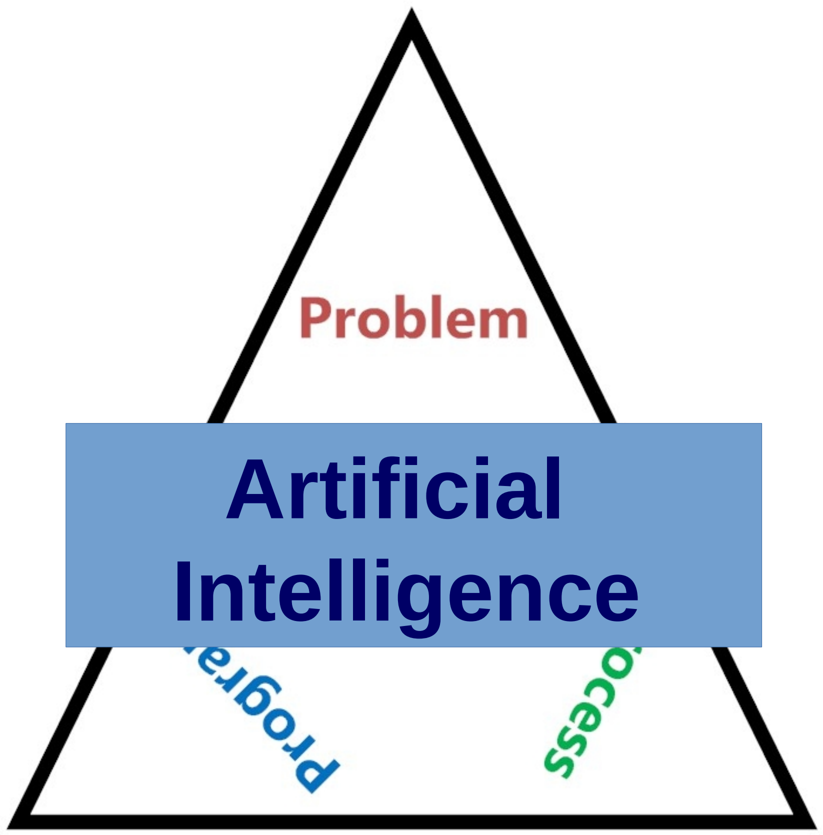
Vector processing

- Systems with several input and/or output signals being computed at once: can work with streams or blocks

Problem Program and Process



Problem Program and Process



Artificial Intelligence

Agenda

- Importance
- Coding Languages
- Conventional Programming
- **Python Language**
- Python for Machine Learning
- Python for Image Processing

Scripting Languages

- Scripting languages focus flexibility, rapid development and dynamic checking.
- Their type systems embrace very high level concepts such as tables, patterns, lists and files.
- There a number of distinct groups that fall under the scripting language family.
- Languages such as Perl and Python are known as ``glue'' languages because they were designed to glue existing programs together.
- There are other extensible types of scripting languages used in the WWW also.

Python Language

Everything in Python is an object.

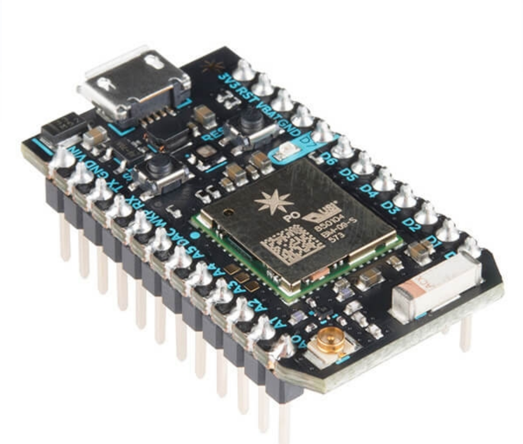
The objects can be either mutable or immutable.

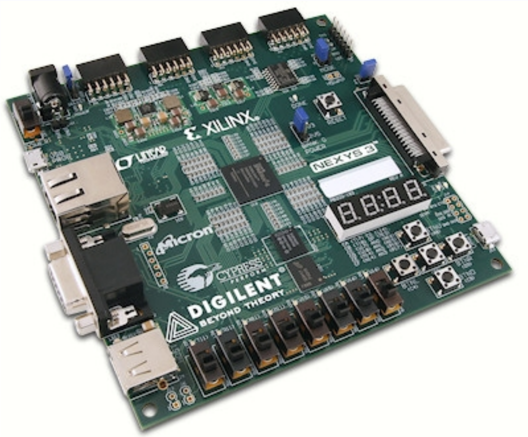
A mutable object can be changed after it is created, and an immutable object can't.

Strengths

- Easy to learn:
- Supports multiple programming paradigms
- Extensible
- Active open source community
- Large and Active Community Support
- Powerful Set of Packages
- Easy and Rapid Prototyping
- Easy to Collaborate

Python for Different Technologies





Courtesy of dvanzuijlekom

Setting Up a Python Environment

- Set Up Anaconda Python Environment
- Installing Libraries
  - pip install required_package
-

UCERD
Gathering Intellectuals
www.ucerd.com

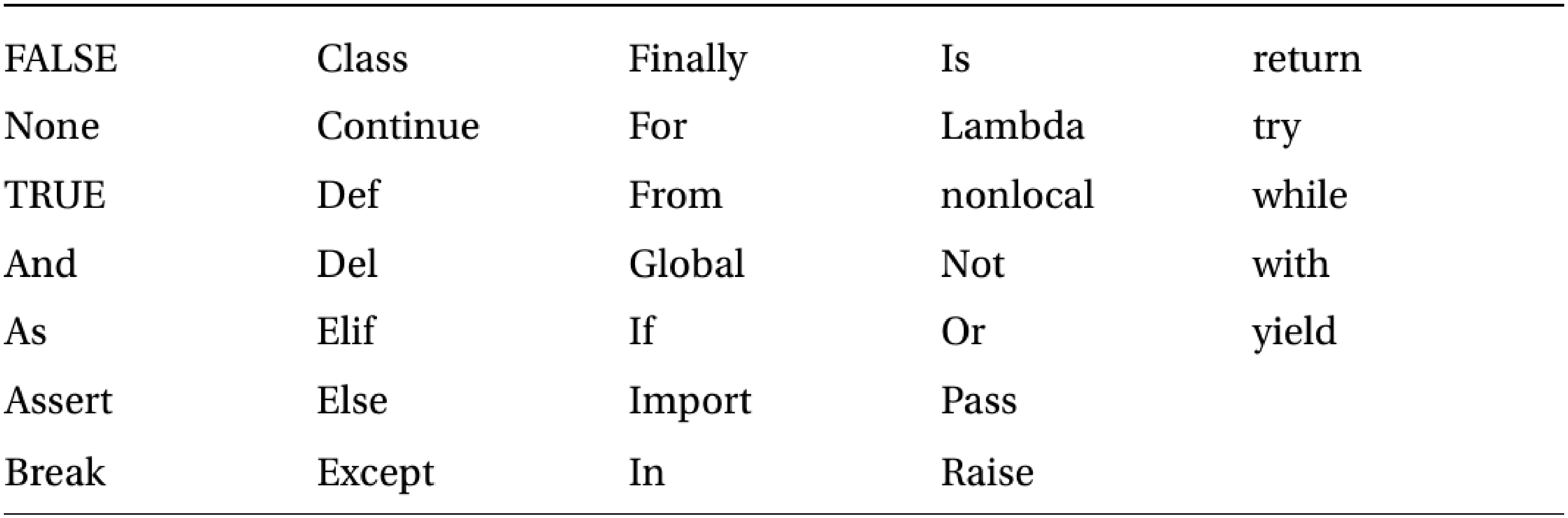
Python for HPC

- Python is known for its very expressive language, easy to read syntax, large community, and impressive range of extension modules.
- Python has regularly come to be used as a program glue due to its ability to interface so easily with external applications.
- With an increase in the availability of Python tools for HPC comes a decrease in the performance barrier between Python and its complied foes.
- There are the more mature projects such as NumPy and SciPy, which offer performance without compilation, as well as recent attempts to bring parallel libraries and just-in-time compiling to Python.
- Numba is a python module that produces compiled code from python functions that can lead to significant speed-ups. Numba is also able to compile code for both Nvidia and AMD GPUs, which presents an exciting new HPC approach consisting of rapid development times and fast execution on desktops.

Development Environments

1. PyDev with Eclipse
2. Komodo
3. Emacs
4. Vim
5. TextMate
6. Gedit
7. Idle
8. PIDA (Linux)(VIM Based)
9. NotePad++ (Windows)
10.BlueFish (Linux)
11. ipython

Python Keywords



FALSE	Class	Finally	Is	return
None	Continue	For	Lambda	try
TRUE	Def	From	nonlocal	while
And	Del	Global	Not	with
As	Elif	If	Or	yield
Assert	Else	Import	Pass	
Break	Except	In	Raise	

Python Scripting Language

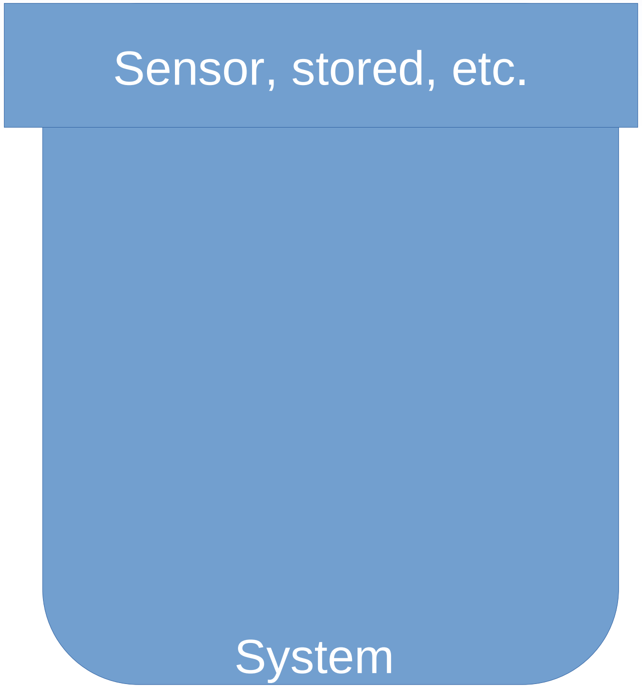
- Data Input Output
- Data Types – Data Structures
- Conditional Statements
- Repetition Statements
- Functions and Libraries

System
Method
Algorithm
.
.

UCERD
Gathering
Intellectuals
www.ucerd.com

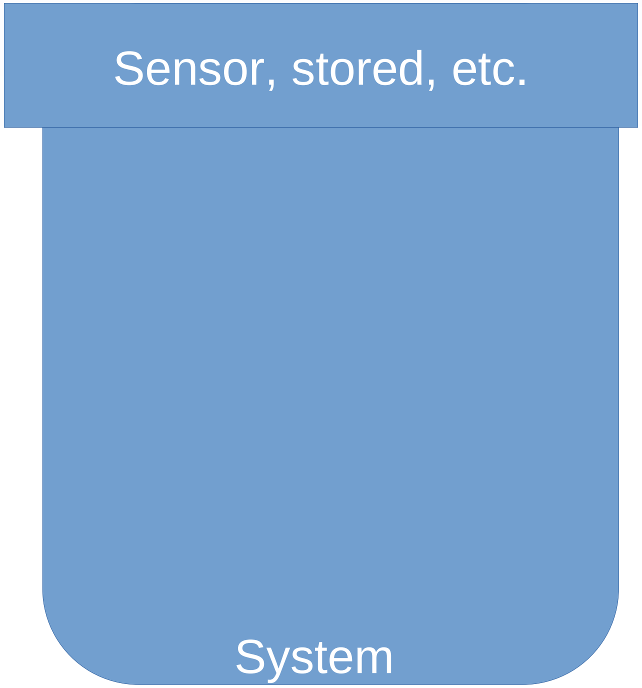
Python Scripting Language

- Data Input Output
- Data Types – Data Structures
- Conditional Statements
- Repetition Statements
- Functions and Libraries



Python Scripting Language

- Data Input Output
- Data Types – Data Structures
- Conditional Statements
- Repetition Statements
- Functions and Libraries

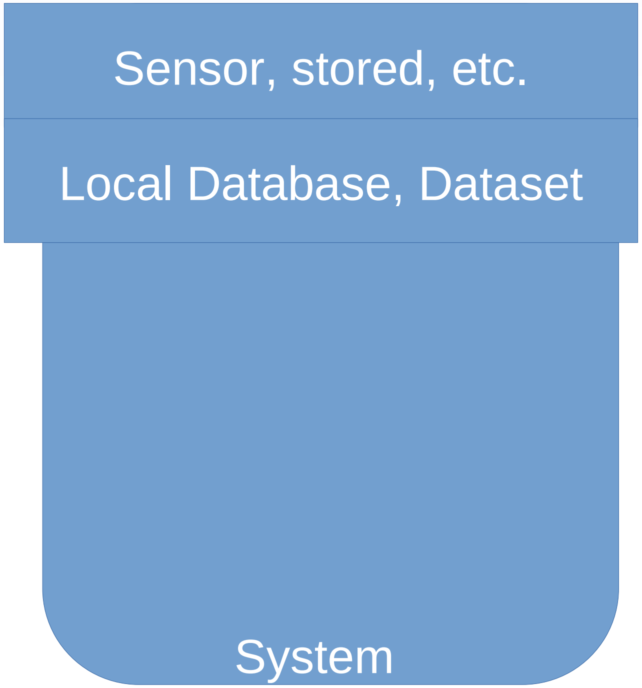


Sensor, stored, etc.

System

Python Scripting Language

- Data Input Output
- **Data Types – Data Structures**
- Conditional Statements
- Repetition Statements
- Functions and Libraries



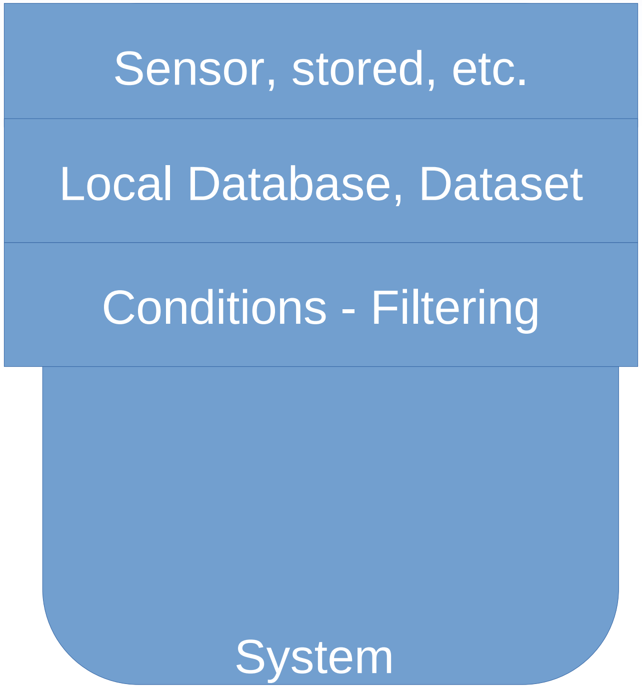
Sensor, stored, etc.

Local Database, Dataset

System

Python Scripting Language

- Data Input Output
- Data Types – Data Structures
- **Conditional Statements**
- Repetition Statements
- Functions and Libraries



Sensor, stored, etc.

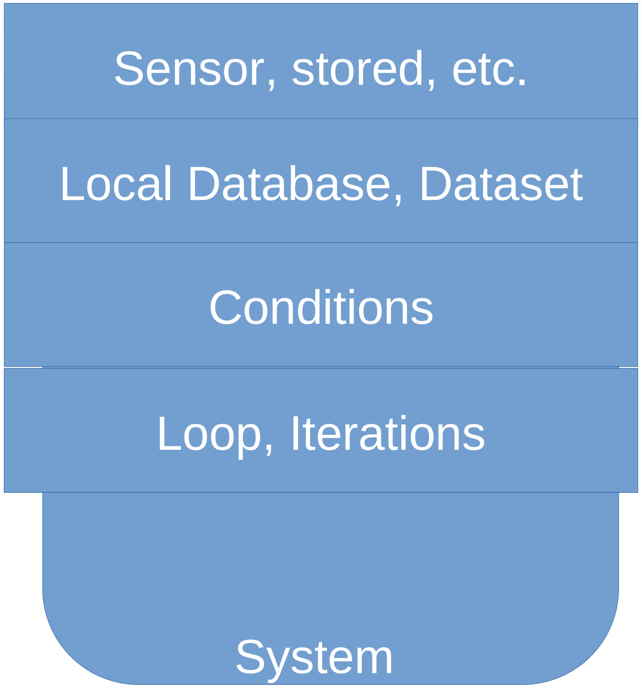
Local Database, Dataset

Conditions - Filtering

System

Python Scripting Language

- Data Input Output
- Data Types – Data Structures
- Conditional Statements
- **Repetition Statements**
- Functions and Libraries



Sensor, stored, etc.

Local Database, Dataset

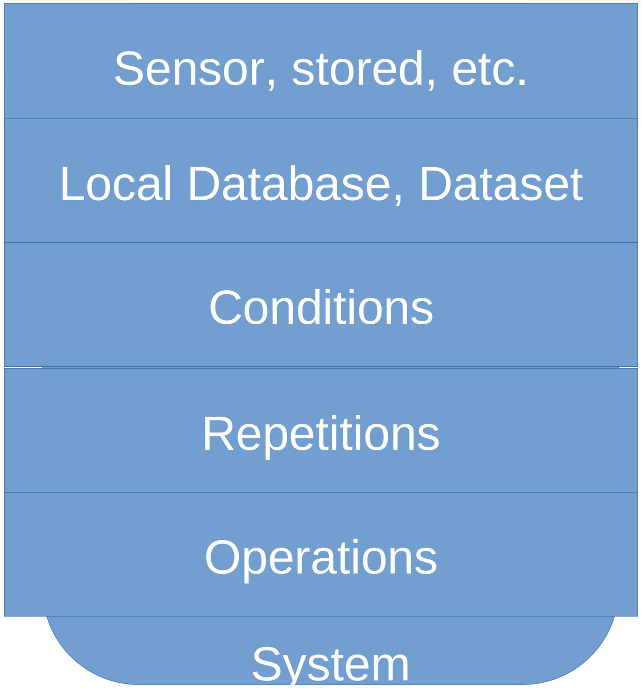
Conditions

Loop, Iterations

System

Python Scripting Language

- Data Input Output
- Data Types – Data Structures
- Conditional Statements
- Repetition Statements
- **Functions and Libraries**



Sensor, stored, etc.

Local Database, Dataset

Conditions

Repetitions

Operations

System

Python Environment

# Libraries

# Read data

# Operations: Filtering, Processing, Classification, Control etc.

# Visualizing

# Write, Operate etc

Modules and Functions

```
import math as mt

mt.functions…


import math
math.cos


from math import cos, pi
cos


from math import *
```

Example 1

Reading and Writing Data

- Understand the source and type of data

Reading Files

```
 f = open(“names.txt")
>>> f.readline()
Results
```

Uses libraries to deal with complex databases and datastructures.

Data Types/Structure

Lists

Tuples

Set

Dictionary

List, Tuple, Set and Dictionary

- List: Use when user need an ordered sequence of homogenous collections, whose values can be changed later in the program.

```
my_list = ['a','b','c','b', 'a']
```

- Tuple: User when you need an ordered sequence of heterogeneous collections whose values need not be changed later in the program.

```
l = (1, 2, 3)
```

- Set: It is ideal for user when user don’t have to store duplicates and is not concerned about the order or the items. User just want to know whether a particular value already exists or not.

```
set([1, 2, 3, 4])
```

- Dictionary: It is ideal for use when user need to relate values with keys, in order to look them up efficiently using a key.

```
d = {'first':'string value', 'second':[1,2]}
d.keys()
```

Condition

```
if continuation :
                    print value



if gpa > 2 :
      print gpa
```

Repetition

```
for x in range(1, 6, +1): # range(start, stop, step)
    print x
```

Agenda

- Importance
- Coding Languages
- Conventional Programming
- Python Language
- **Python for Machine Learning**
- Python for Image Processing

Numpy

- NumPy, short for Numerical Python, is the foundational package for scientific computing in Python.
- Numpy is the backbone of Machine Learning in Python. It is one of the most important libraries in Python for numerical computations. It adds support to core Python for multi-dimensional arrays (and matrices) and fast vectorized operations on these arrays.

Numpy

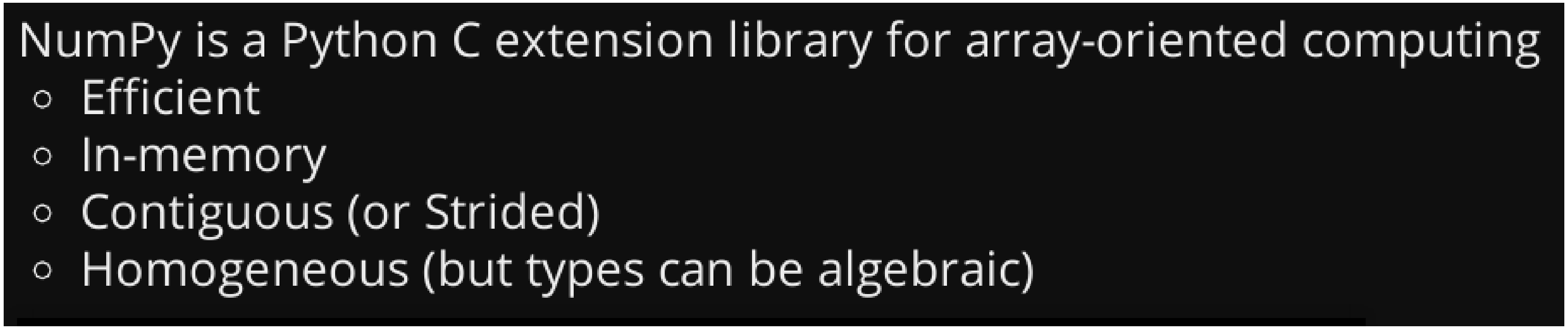
- A fast and efficient **multidimensional array** object ndarray
- **Functions** for performing element-wise computations with arrays or mathematical operations between arrays
- Tools for **reading and writing array-based** data sets to disk
- **Linear algebra operations, Fourier transform, and random number generation**
- Tools for integrating **connecting** C, C++, and Fortran code to Python

Module: Numpy

NumPy is the fundamental package for scientific computing with Python.

It contains among other things:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

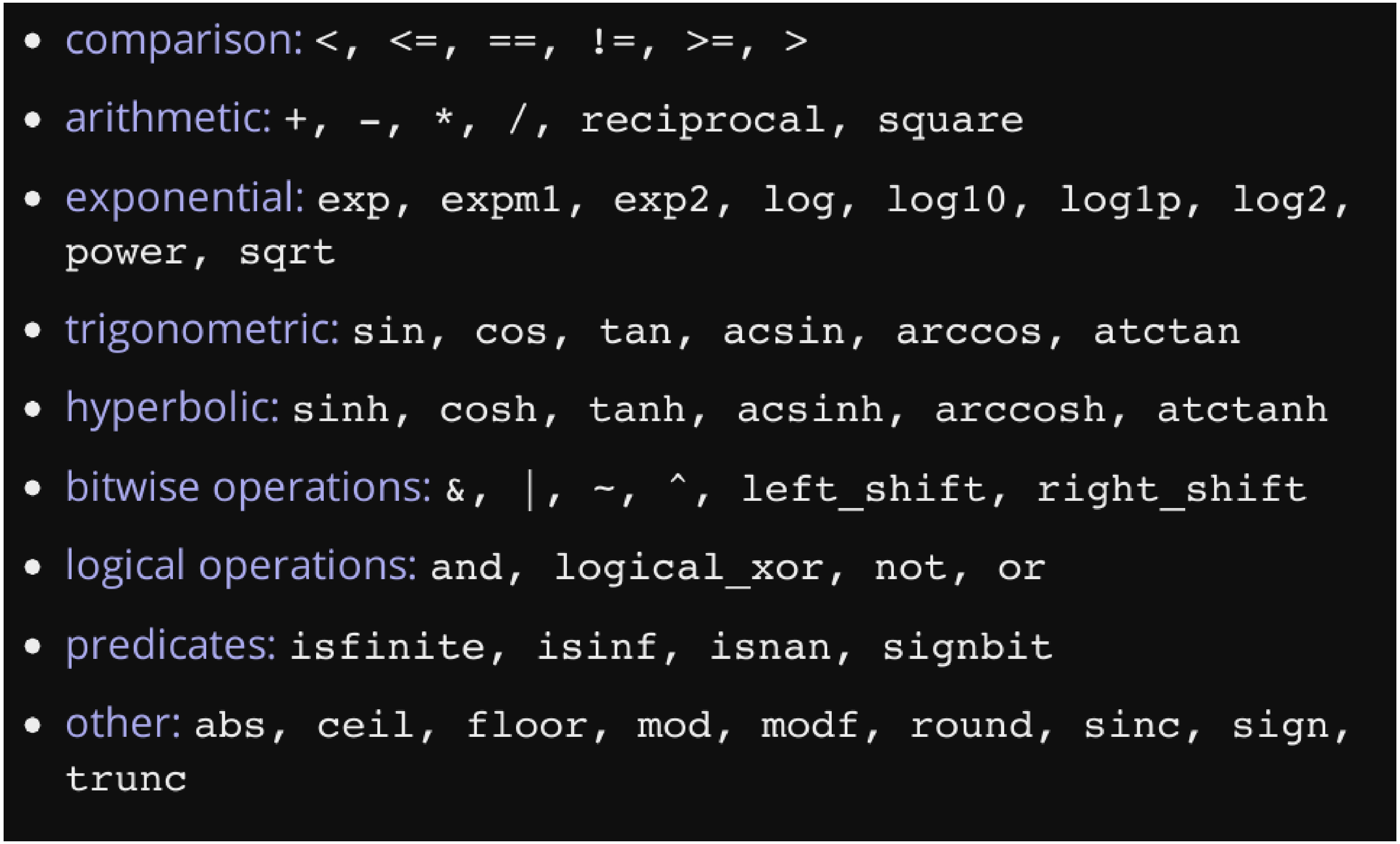


NumPy is a Python C extension library for array-oriented computing

- Efficient
- In-memory
- Contiguous (or Strided)
- Homogeneous (but types can be algebraic)

Source: http://www.numpy.org/

NumPy Functions



- comparison: `<, <=, ==, !=, >=, >`
- arithmetic: `+, -, *, /, reciprocal, square`
- exponential: `exp, expm1, exp2, log, log10, log1p, log2, power, sqrt`
- trigonometric: `sin, cos, tan, acsin, arccos, atctan`
- hyperbolic: `sinh, cosh, tanh, acsinh, arccosh, atctanh`
- bitwise operations: `&, |, ~, ^, left_shift, right_shift`
- logical operations: `and, logical_xor, not, or`
- predicates: `isfinite, isinf, isnan, signbit`
- other: `abs, ceil, floor, mod, modf, round, sinc, sign, trunc`

Pandas

Pandas provides rich data structures and functions designed to make working with structured data fast, easy, and expressive.

Pandas

Pandas is an important Python library for data manipulation, wrangling, and analysis.

Pandas allows you to work with both cross-sectional data and time series based data. So let’s get started exploring pandas!

All the data representation in pandas is done using two primary data structures:

- Series
- Dataframes

Matplotlib

matplotlib is the most popular Python library for producing plots and other 2D data visualizations.

It is well-suited for creating plots suitable for publication.

Provides a comfortable interactive environment for plotting and exploring data.

SciPy

SciPy is a collection of packages addressing a number of different standard problem domains in scientific computing.

Files

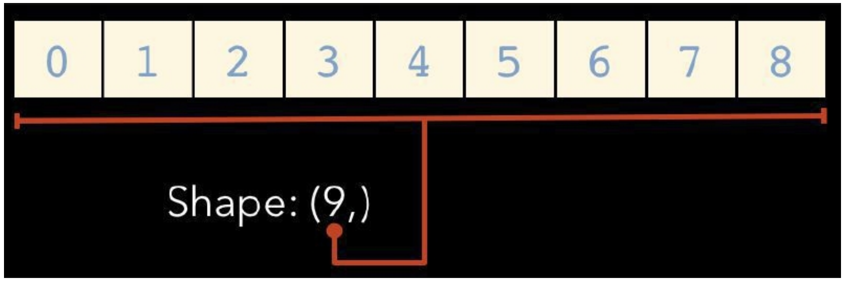
List of dictionaries

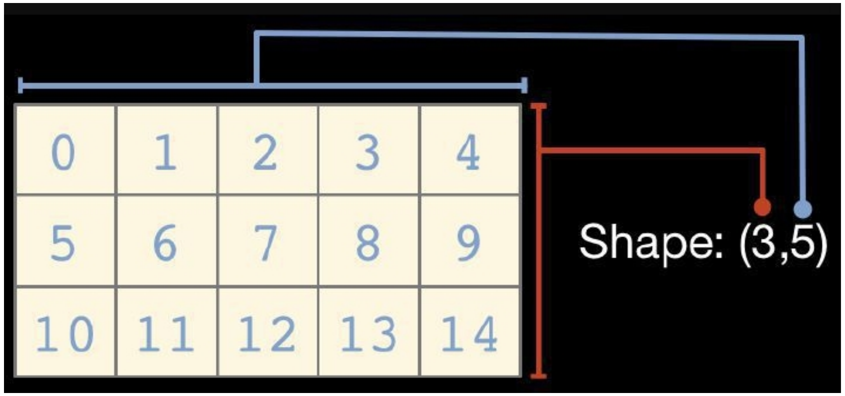
CSV files

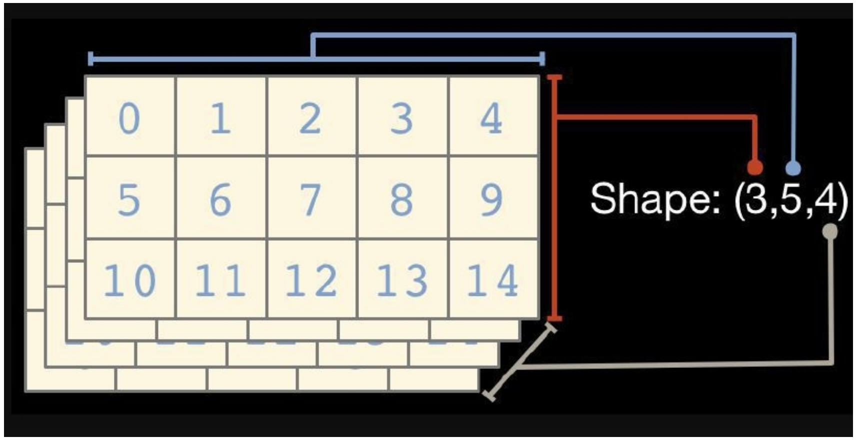
Databases

Understand Dataset

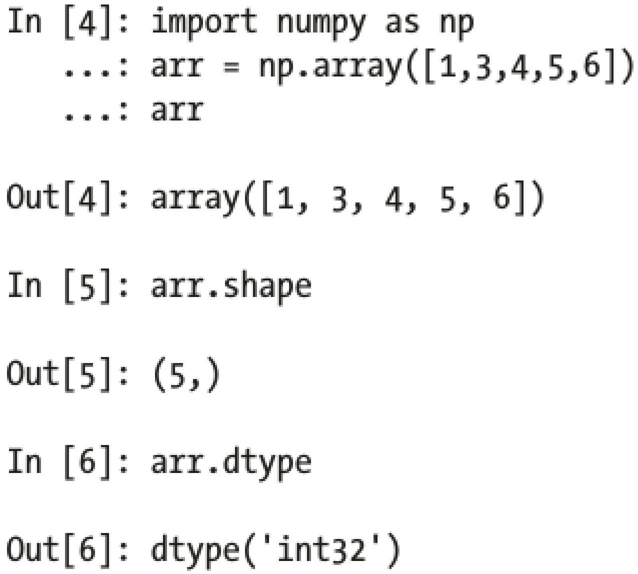
```
data.dtype

np.size(data)

np.shape(data)
```


0 1 2 3 4 5 6 7 8
Shape: (9,)

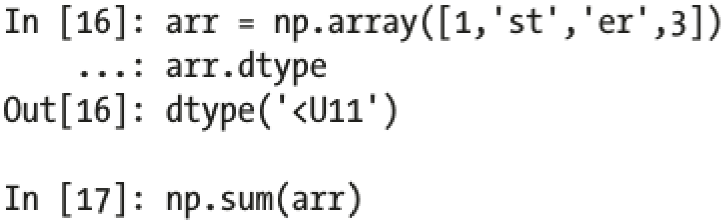

0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
Shape: (3,5)


0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
Shape: (3,5,4)

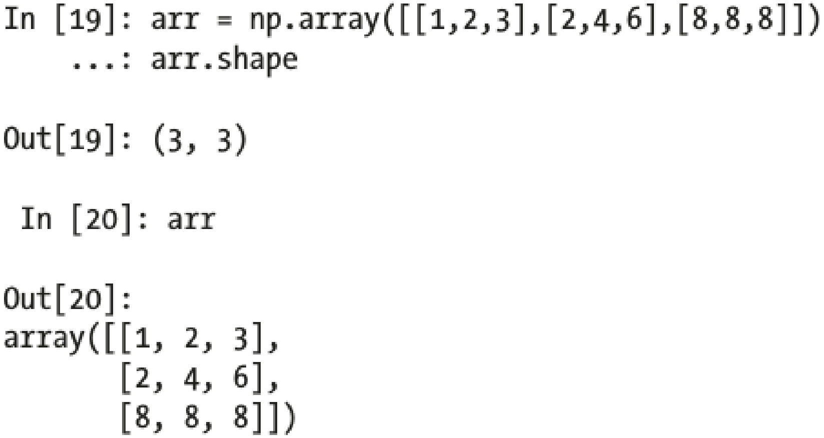
```
data2 = genfromtxt('./signals/ecg.csv',
   delimiter=',')
np.shape(data2)

Signal = data2[:,1]
```



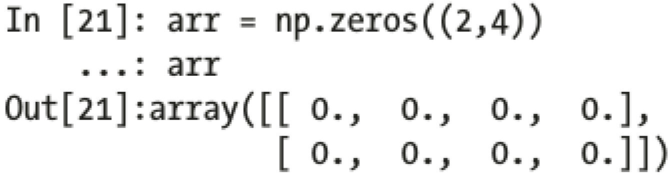
```
In [4]: import numpy as np
   ...: arr = np.array([1,3,4,5,6])
   ...: arr

Out[4]: array([1, 3, 4, 5, 6])

In [5]: arr.shape

Out[5]: (5,)

In [6]: arr.dtype

Out[6]: dtype('int32')
```



```
In [16]: arr = np.array([1,'st','er',3])
    ...: arr.dtype
Out[16]: dtype('<U11')

In [17]: np.sum(arr)
```

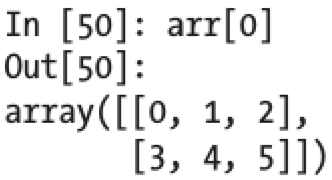


```
In [19]: arr = np.array([[1,2,3],[2,4,6],[8,8,8]])
    ...: arr.shape

Out[19]: (3, 3)

 In [20]: arr

Out[20]:
array([[1, 2, 3],
       [2, 4, 6],
       [8, 8, 8]])
```

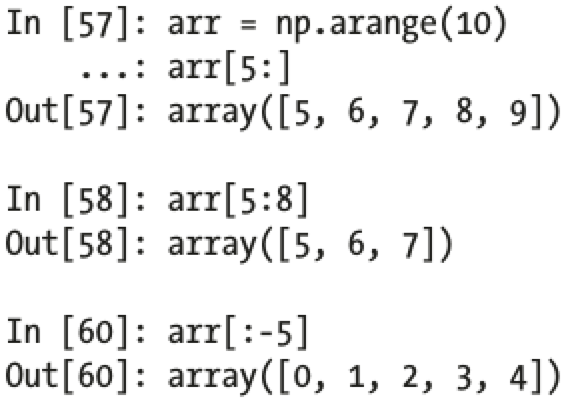


```
In [21]: arr = np.zeros((2,4))
    ...: arr
Out[21]:array([[ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.]])
```

Accessing Array Element

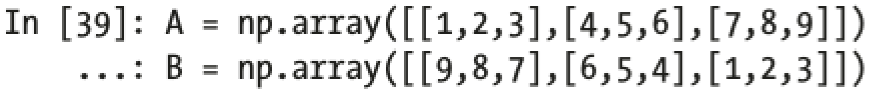


```
In [50]: arr[0]
Out[50]:
array([[0, 1, 2],
       [3, 4, 5]])
```

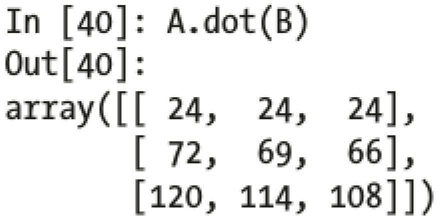


```
In [57]: arr = np.arange(10)
    ...: arr[5:]
Out[57]: array([5, 6, 7, 8, 9])

In [58]: arr[5:8]
Out[58]: array([5, 6, 7])

In [60]: arr[:-5]
Out[60]: array([0, 1, 2, 3, 4])
```

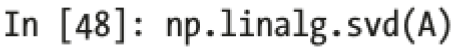
Linear Algebra Using numpy



```
In [39]: A = np.array([[1,2,3],[4,5,6],[7,8,9]])
    ...: B = np.array([[9,8,7],[6,5,4],[1,2,3]])
```

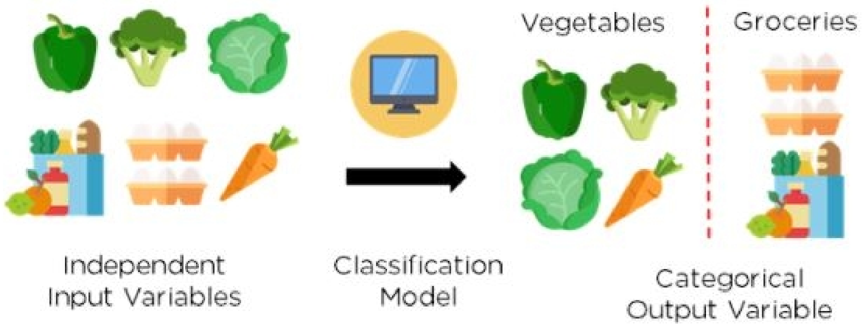


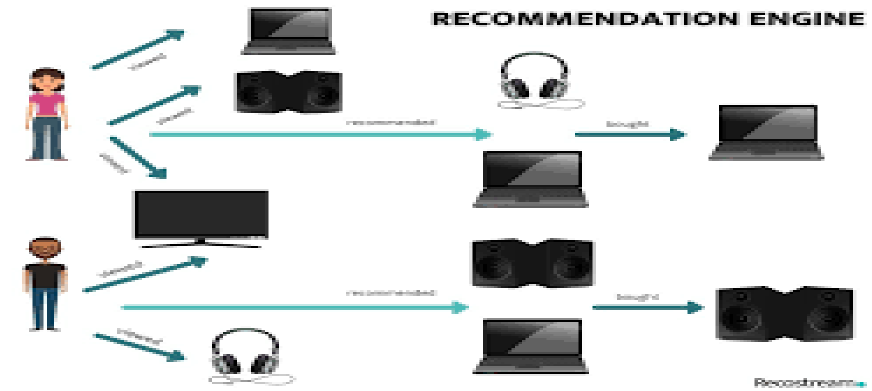
```
In [40]: A.dot(B)
Out[40]:
array([[ 24,  24,  24],
       [ 72,  69,  66],
       [120, 114, 108]])
```



```
In [48]: np.linalg.svd(A)
```

Machine Learning





Classification

Categorization

Prediction

Anomaly Detection

Regression

Recommendation Systems

Natural Language Processing (NLP)

Time Series Forecasting

Clustering and Segmentation

Reinforcement Learning

Image Generation

Image Processing Toolkit OpenCV

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Source: https://opencv.org/

Read and Display

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

#input handler
img = cv2.imread('./images/L1.jpg')
plt.imshow(img)
plt.show()
```

OpenCV

Intel® OPEN SOURCE COMPUTER VISION LIBRARY

Goals

- Develop a universal toolbox for research and development in the field of Computer Vision

OpenCV Functionality (more than 350 algorithms)

- **Basic structures and operations**
- **Image Analysis**
- **Structural Analysis**
- Object Recognition
- Motion Analysis and Object Tracking
- 3D Reconstruction

Basic Structures and Operations

- Image and Video Data Structures

```
Mat image;
Image = imread (“path”);
```

- Multidimensional array operations

include operations on images, matrices and histograms.

```
equalizeHist( src, dst );
```

- Dynamic structures operations

concern all vector data storages.

- Drawing primitives

allows not only to draw primitives but to use the algorithms for pixel access

- Utility functions

in particular, contain fast implementations of useful math functions.

Image Read/Write

- Import cv2 as cv

```
gray_img = cv2.imread('images/input.jpg', cv2.IMREAD_GRAYSCALE)
cv2.imshow('Grayscale', gray_img)
cv2.waitKey()

cv2.imwrite('images/output.jpg', gray_img)

gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
yuv_img = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
```

Image Analysis

- Thresholds

  threshold(src_gray, dst, threshold_value, max_BINARY_value, threshold_type);

- Statistics
- Pyramids
- Morphology

  Erosion , dilation etc

- Distance transform
- Feature detection

UCERD
Gathering
Intellectuals
www.ucerd.com

Statistics

- min, max, mean value, standard deviation over the image
- Norms C, L1, L2
- Multidimensional histograms
- Spatial moments up to order 3 (central, normalized, Hu)

Pyramids

An image pyramid is a collection of images - all arising from a single original image - that are successively downsampled until some desired stopping point is reached.

PyrUp()

pyrdown()

Gaussian pyramid:

Laplacian pyramid:

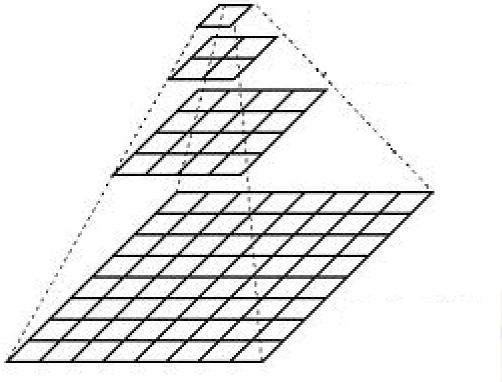
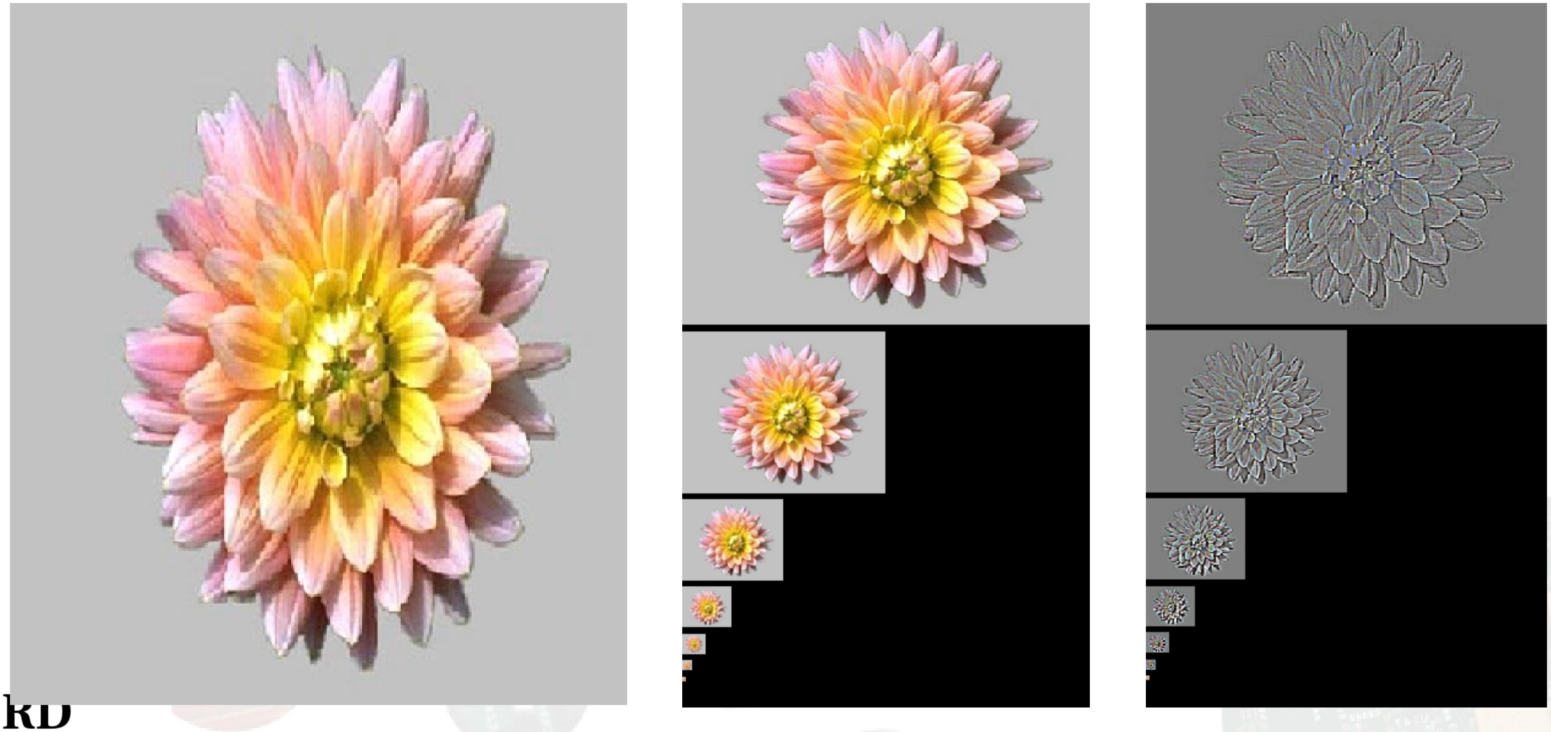


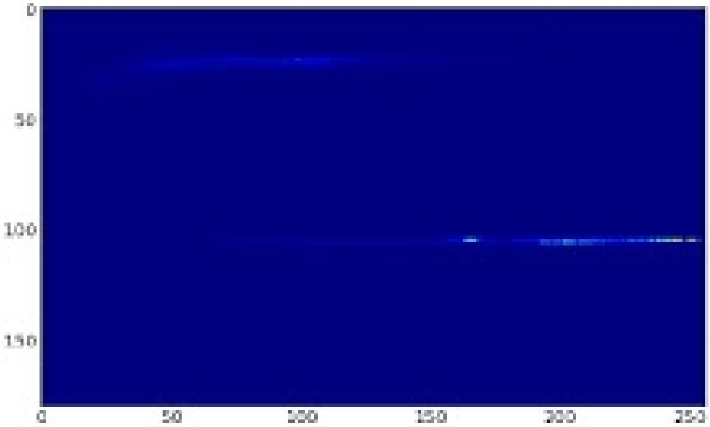
Image Pyramids

- Gaussian and Laplacian



Multidimensional Histograms

- Histogram operations : calculation, normalization, comparison, back project
- Histograms types:
  - ✓ Dense histograms
  - ✓ Signatures (balanced tree)



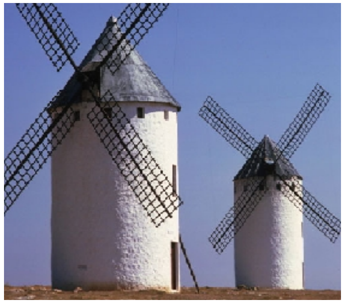
Morphological Operations

- Two basic morphology operations using structuring element:
  - ✓ erosion
  - ✓ dilation
- More complex morphology operations:
  - ✓ opening
  - ✓ closing
  - ✓ morphological gradient
  - ✓ top hat
  - ✓ black hat

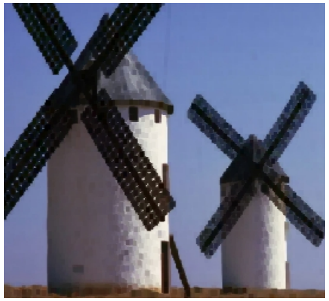
Morphological Operations Examples

- Morphology - applying Min-Max. Filters and its combinations

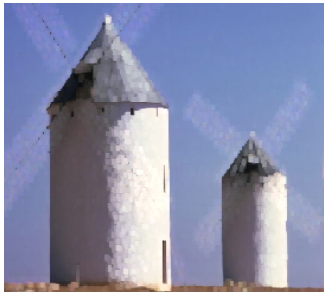
Image I



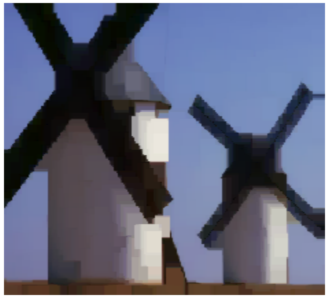
Erosion IΘB



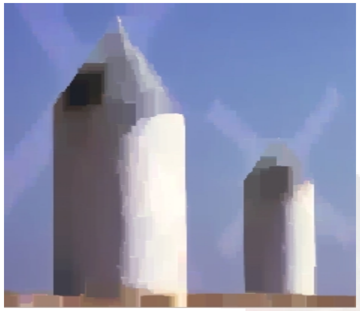
Dilatation I⊕B



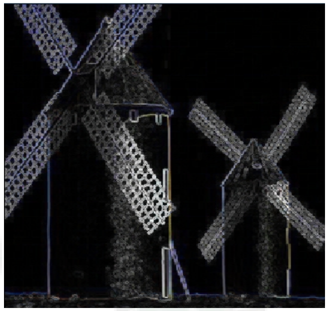
Opening IoB= (IΘB)⊕B



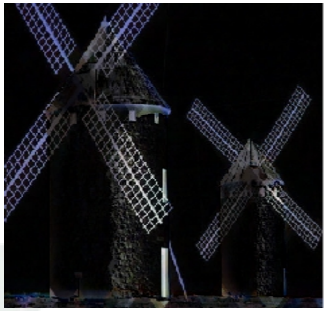
Closing I•B= (I⊕B)ΘB



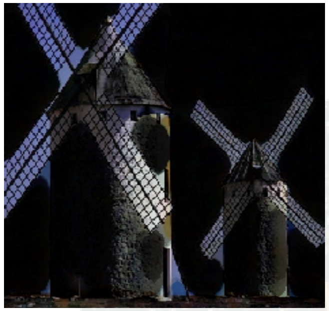
Grad(I)= (I⊕B)-(IΘB)



TopHat(I)= I - (IΘB)



BlackHat(I)= (I⊕B) - I

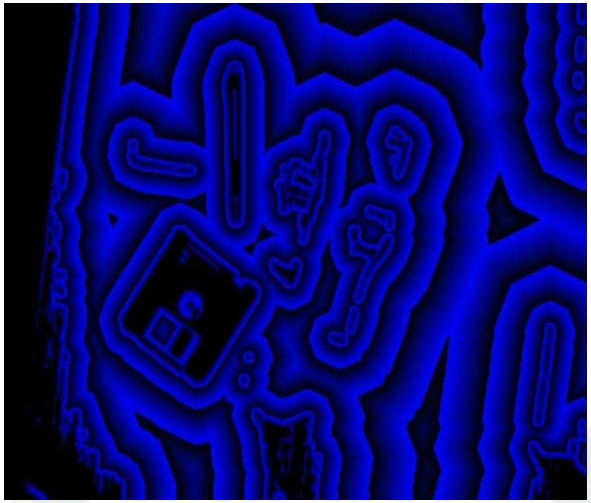


Distance Transform

The distance transform operator generally takes binary images as inputs. In this operation, the gray level intensities of the points inside the foreground regions are changed to distance their respective distances from the closest 0 value (boundary).

distanceTransform()

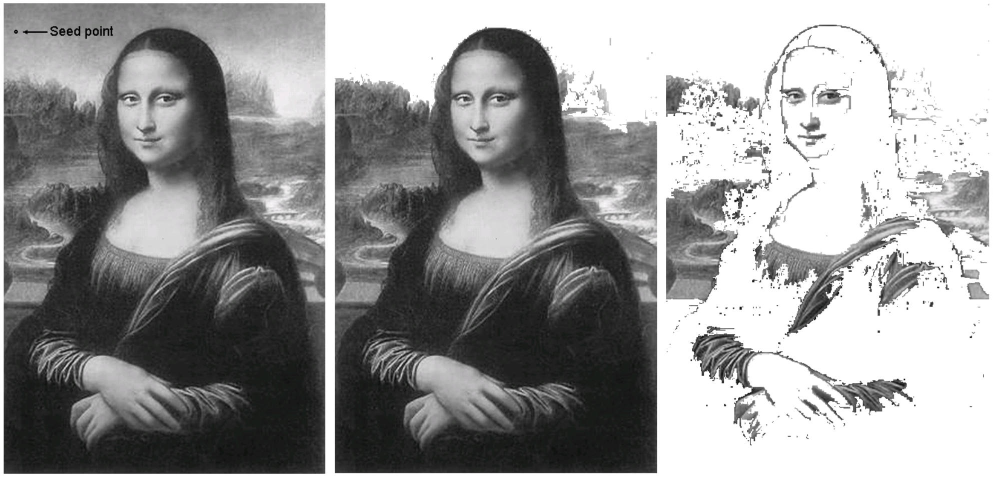
- Calculate the distance for all non-feature points to the closest feature point
- Two-pass algorithm, 3x3 and 5x5 masks, various metrics predefined



Flood Filling

- Simple
- Gradient

```
cv2.floodFill(img, mask, (0,0), 255);
```



Original image Tolerance interval ± 5 Tolerance interval ± 6

IN GOD WE TRUST
LIBERTY · 2004

Feature Detection

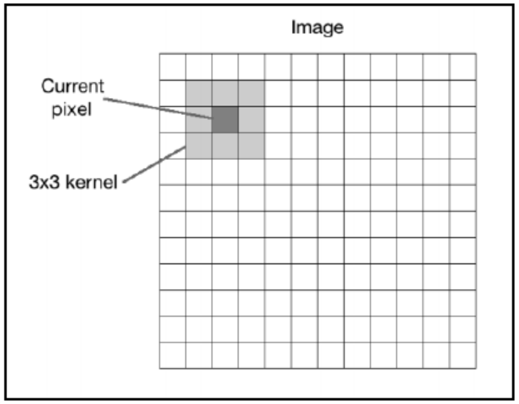
- Fixed filters (Sobel operator, Laplacian);
- Optimal filter kernels with floating point coefficients (first, second derivatives, Laplacian)
- Special feature detection (corners)
- Canny operator
- Hough transform (find lines and line segments)
- Gradient runs

Convolution

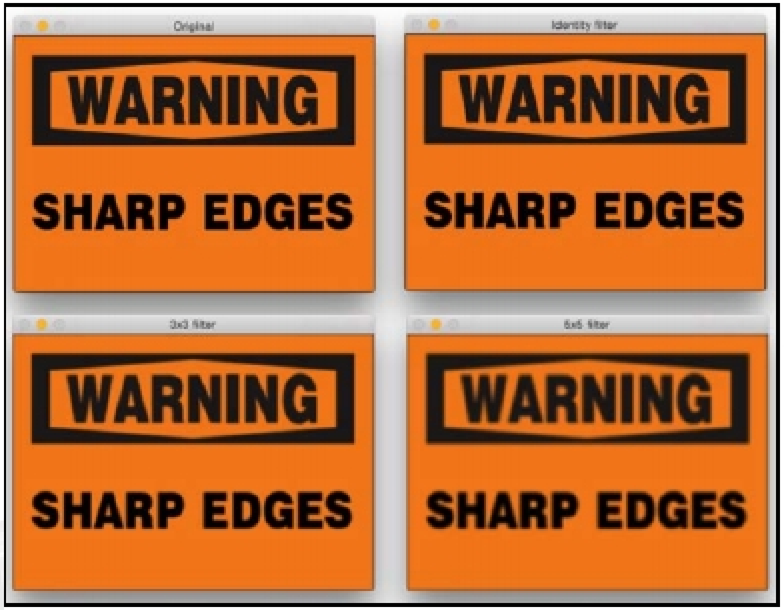
Convolution is a fundamental operation in image processing. It basically applies a mathematical operator to each pixel, and change its value in some way.

To apply this mathematical operator, convolution uses another matrix called a kernel. The kernel is usually much smaller in size than the input image. For each pixel in the image, we take the kernel and place it on top so that the center of the kernel coincides with the pixel under consideration.

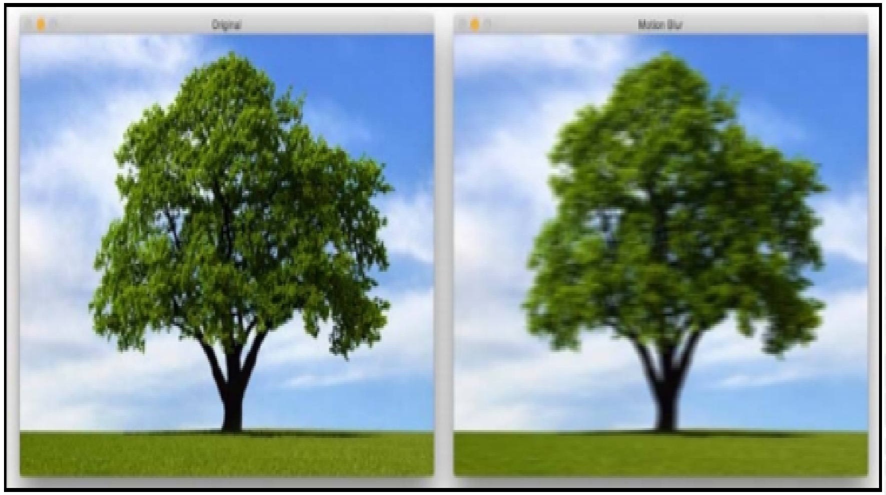
We then multiply each value in the kernel matrix with the corresponding values in the image, and then sum it up. This is the new value that will be applied to this position in the output image.



```
import cv2
import numpy as np
img = cv2.imread('images/input.jpg')
rows, cols = img.shape[:2]
kernel_identity = np.array([[0,0,0], [0,1,0], [0,0,0]])
kernel_3x3 = np.ones((3,3), np.float32) / 9.0 # Divide by 9 to normalize the kernel
kernel_5x5 = np.ones((5,5), np.float32) / 25.0 # Divide by 25 to normalize the kernel
cv2.imshow('Original', img)
# value -1 is to maintain source image depth
output = cv2.filter2D(img, -1, kernel_identity)
cv2.imshow('Identity filter', output)
output = cv2.filter2D(img, -1, kernel_3x3)
cv2.imshow('3x3 filter', output)
output = cv2.filter2D(img, -1, kernel_5x5)
cv2.imshow('5x5 filter', output)
cv2.waitKey(0)
```



```
import cv2
from matplotlib import pyplot as plt
import numpy as np
img = cv2.imread('images/input.jpg')
cv2.imshow('Original', img)
size = 15
# generating the kernel
kernel_motion_blur = np.zeros((size, size))
kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
kernel_motion_blur = kernel_motion_blur / size
# applying the kernel to the input image
output = cv2.filter2D(img, -1, kernel_motion_blur)
cv2.imshow('Motion Blur', output)
cv2.waitKey(0)
```

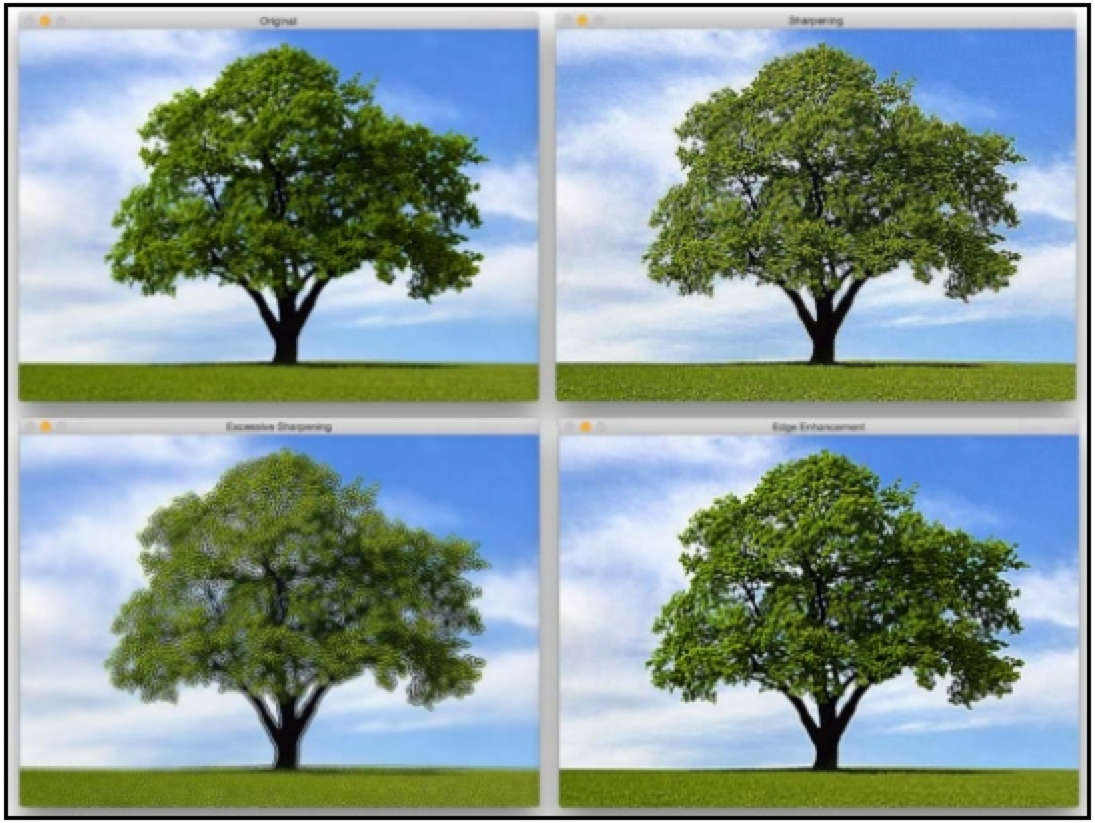


Sharpening Images

```
# generating the kernels
kernel_sharpen_1 = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
kernel_sharpen_2  = np.array([[1,1,1], [1,-7,1], [1,1,1]])
kernel_sharpen_3  = np.array([[-1,-1,-1,-1,-1], [-1,2,2,2,-1], [-1,2,8,2,-1], [-1,2,2,2,-1], [-1,-1,-1,-1,-1]]) / 8.0
```

$$M = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -7 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

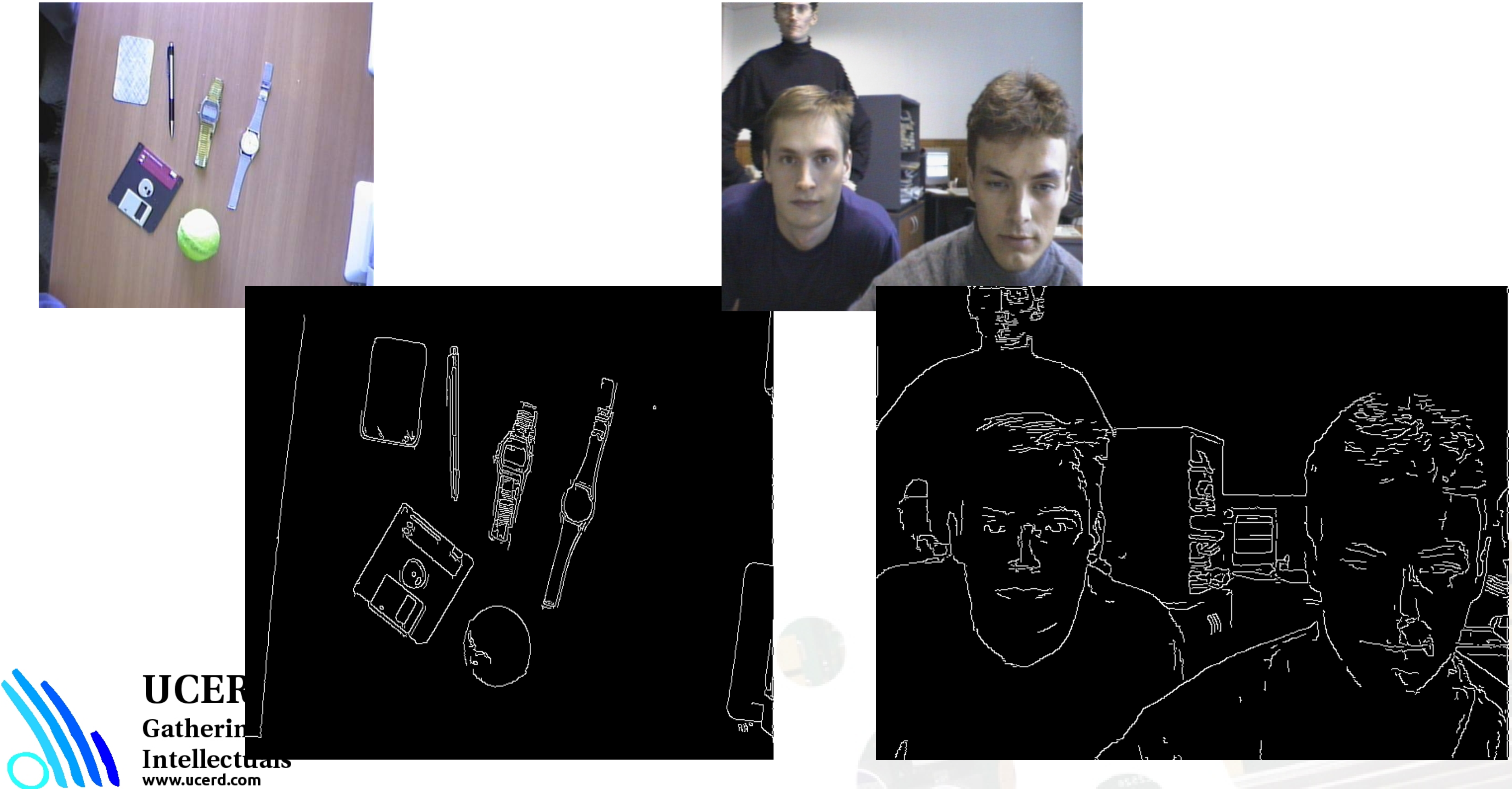


The process of edge detection involves detecting sharp edges in the image, and producing a binary image as the output. Typically, we draw white lines on a black background to indicate those edges.

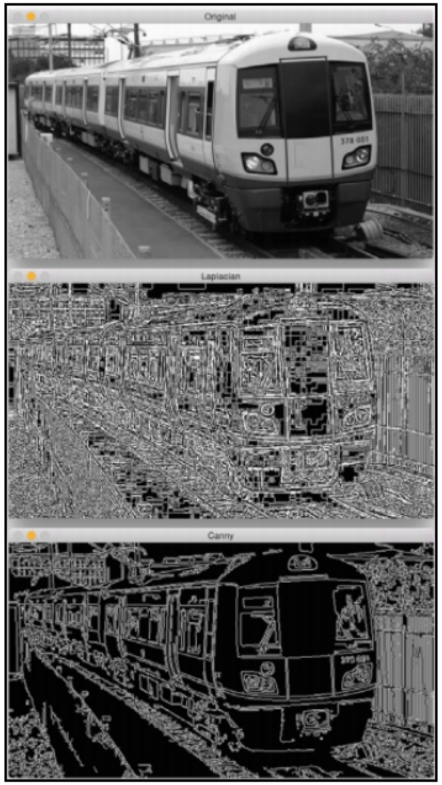
We can think of edge detection as a high pass filtering operation. A high pass filter allows high-frequency content to pass through and blocks the low-frequency content. As we discussed earlier, edges are high-frequency content. In edge detection, we want to retain these edges and discard everything else. Hence, we should build a kernel that is the equivalent of a high pass filter.

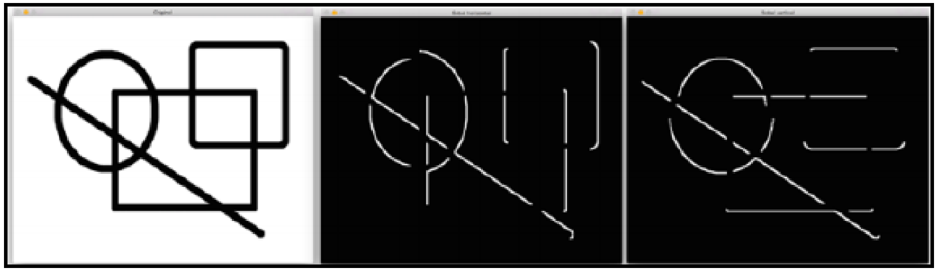
$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Canny Edge Detector



```
import cv2
import numpy as np
img = cv2.imread('images/input_shapes.png',
cv2.IMREAD_GRAYSCALE)
rows, cols = img.shape # It is used depth of
cv2.CV_64F.
sobel_horizontal = cv2.Sobel(img, cv2.CV_64F, 1, 0,
ksize=5)
# Kernel size can be: 1,3,5 or 7.
sobel_vertical = cv2.Sobel(img, cv2.CV_64F, 0, 1,
ksize=5)
cv2.imshow('Original', img)
cv2.imshow('Sobel horizontal', sobel_horizontal)
cv2.imshow('Sobel vertical', sobel_vertical)
cv2.waitKey(0)
```



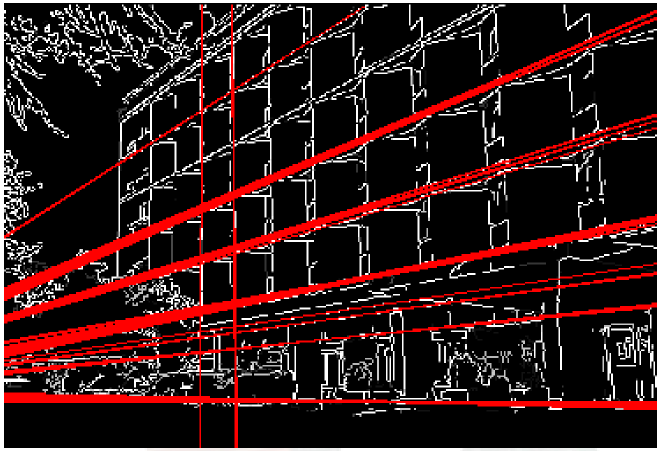


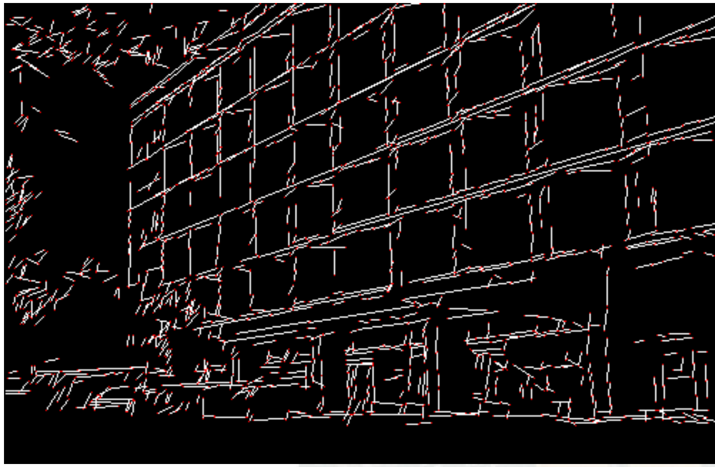
Hough Transform

Detects lines in a binary image

- Standard Hough Transform

- Probabilistic Hough Transform





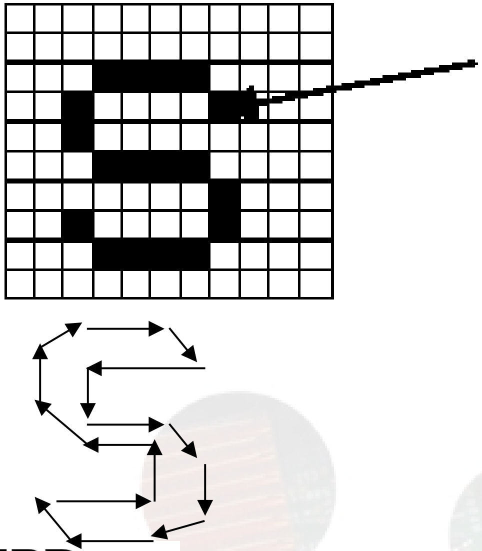
Hough Transform

Detects lines in a binary image

Hough Transform is a popular technique to detect any shape, if you can represent that shape in mathematical form. It can detect the shape even if it is broken or distorted a little bit. We will see how it works for a line.

Contour Retrieving

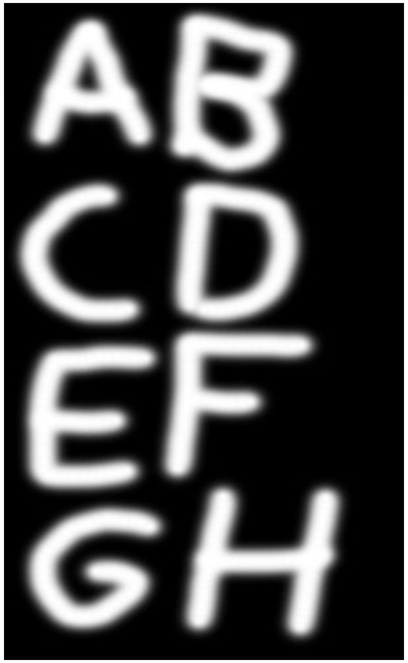
- The contour representation:
  - ✓ Chain code (Freeman code)
  - ✓ Polygonal representation



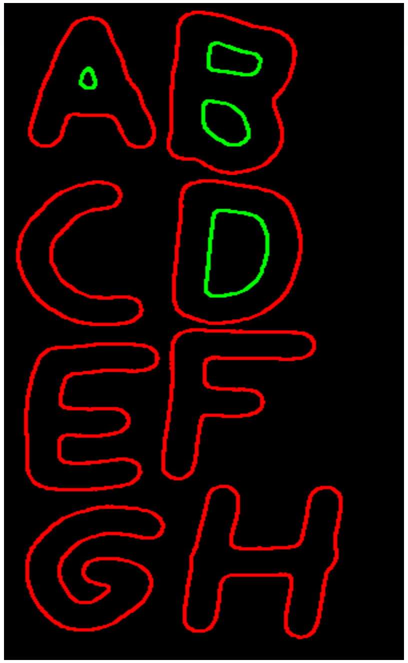
Initial Point

Chain code for the curve:
34445670007654443

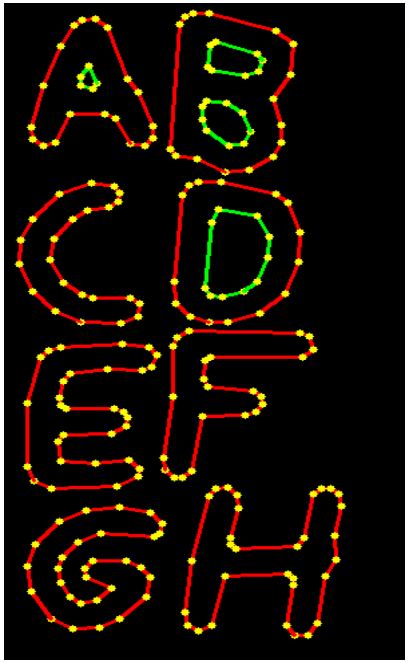
Contours Examples



Source Picture
(300x600 = 180000 pts total)



Retrieved Contours
(<1800 pts total)



After Approximation
(<180 pts total)

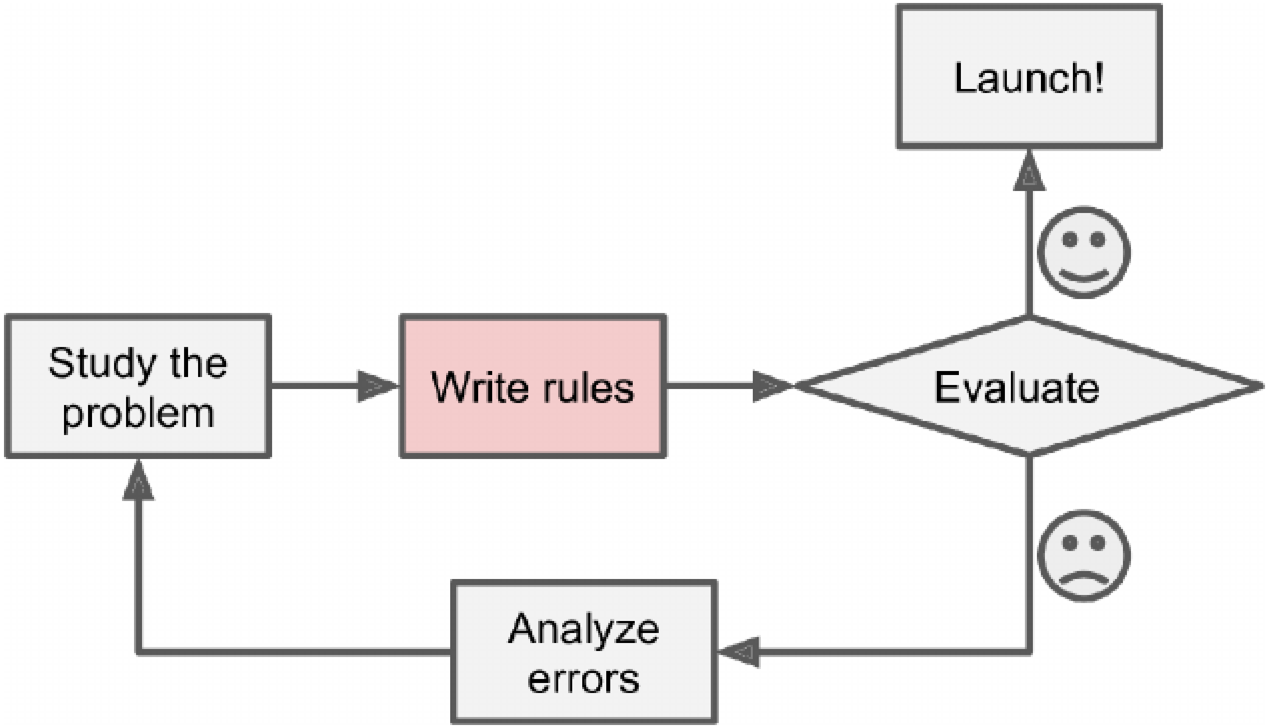
And it is rather fast: ~70 FPS for 640x480 on complex scenes

Machine Learning

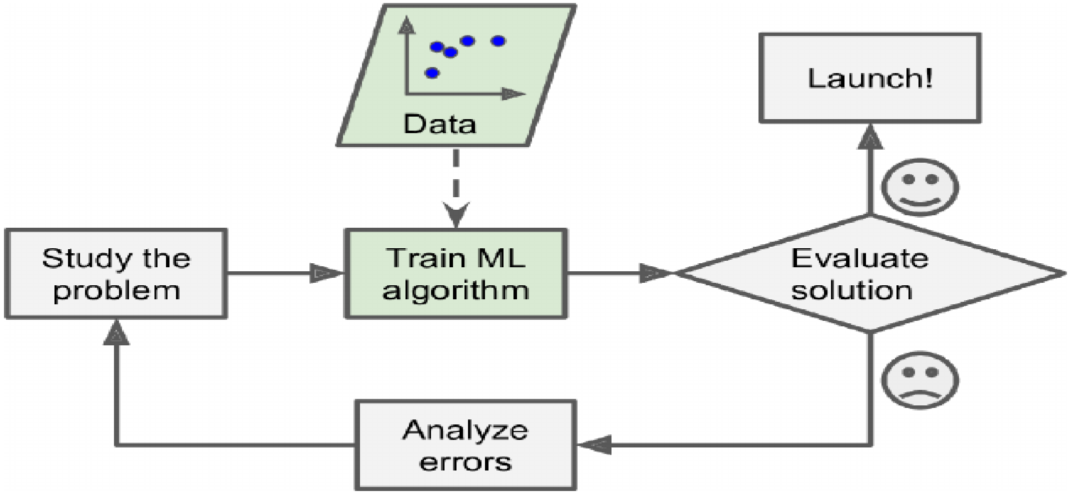
Example

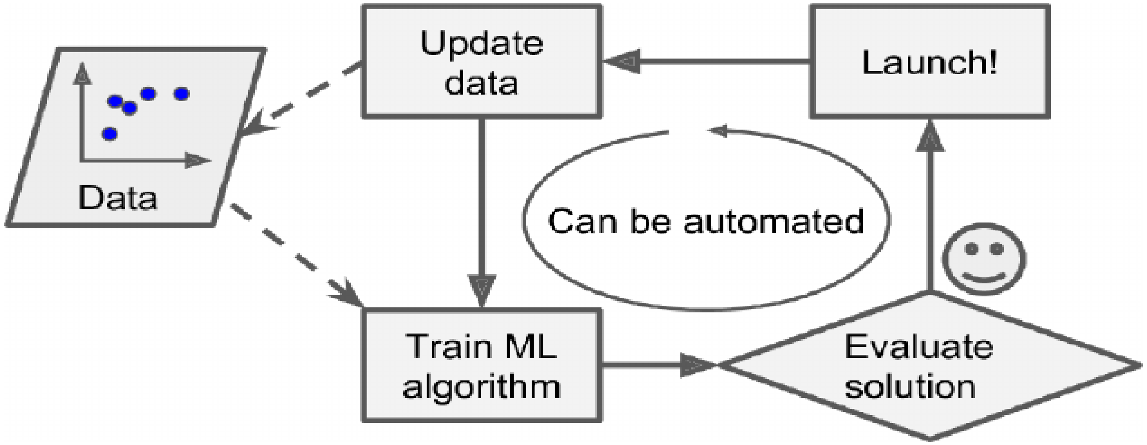
Processor / System	Dhrystone MIPS / MIPS
Nios II	190 MIPS at 165 MHz
ARM Cortex A7	2,850 MIPS at 1.5 GHz
ARM Cortex-A9 (Dual core)	7,500 MIPS at 1.5 GHz
Raspberry Pi 2	1186 MIPS per core at 1.0 GHz
Nvidia Tegra 3 (Quad core Cortex-A9)	13,800 MIPS at 1.5 GHz
Intel Core 2 Extreme QX6700 (Quad core)	49,161 MIPS at 2.66 GHz

Conventional Approach Write an Application

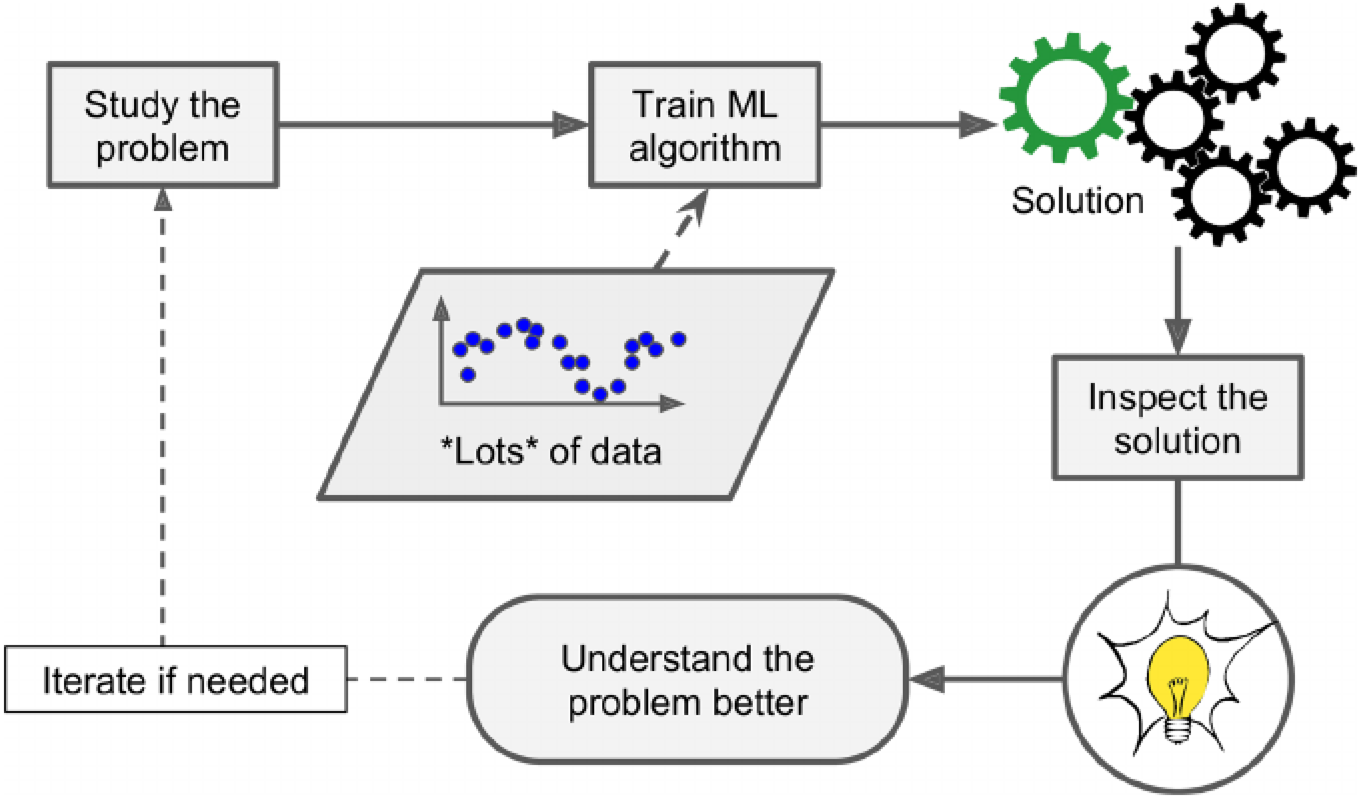


ML Approach Write an Application





UCERD
Gathering
Intellectuals
www.ucerd.com


Study the problem
Train ML algorithm
Solution
Lots of data
Inspect the solution
Understand the problem better
Iterate if needed

Types of Machine Learning Systems

- Whether or not they are trained with human supervision (supervised, unsupervised, semisupervised, and Reinforcement Learning)
- Whether or not they can learn incrementally on the fly (online versus batch learning)
- Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do (instance-based versus model-based learning)

Analyzing images of products on a production line to automatically classify them: This is image classification, typically performed using convolutional neural net-works (CNNs; see Chapter 14).

Detecting tumors in brain scans: This is semantic segmentation, where each pixel in the image is classified (as we want to determine the exact location and shape of tumors), typically using CNNs as well.

Automatically classifying news articles:

This is natural language processing (NLP), and more specifically text classification, which can be tackled using recurrent neural networks (RNNs), CNNs, or Transformers

Automatically flagging offensive comments on discussion forums: This is also text classification, using the same NLP tools.

Summarizing long documents automatically: This is a branch of NLP called text summarization, again using the same tools.

Creating a chatbot or a personal assistant: This involves many NLP components, including natural language understanding (NLU) and question-answering modules.

Forecasting your company's revenue next year, based on many performance metrics This is a regression task (i.e., predicting values) that may be tackled using any regression model, such as a Linear Regression or Polynomial Regression model (see Chapter 4), a regression SVM (see Chapter 5), a regression Random Forest (see Chapter 7), or an artificial neural network (see Chapter 10). If you want to take into account sequences of past performance metrics, you may want to use RNNs, CNNs, or Transformers (see Chapters 15 and 16).

Making your app react to voice commands:

This is speech recognition, which requires processing audio samples: since they are long and complex sequences, they are typically processed using RNNs, CNNs, or Transformers (see Chapters 15 and 16).

Detecting credit card fraud: This is anomaly detection (see Chapter 9).

Segmenting clients based on their purchases so that you can design a different marketing: strategy for each segment This is clustering (see Chapter 9).

Representing a complex, high-dimensional dataset in a clear and insightful diagram: This is data visualization, often involving dimensionality reduction techniques

Recommending a product that a client may be interested in, based on past purchases: This is a recommender system. One approach is to feed past purchases (and other information about the client) to an artificial neural network.

Building an intelligent bot for a game: This is often tackled using Reinforcement Learning (RL; see Chapter 18), which is a branch of Machine Learning that trains agents (such as bots) to pick the actions that will maximize their rewards over time (e.g., a bot may get a reward every time the player loses some life points), within a given environment (such as the game). The famous AlphaGo program that beat the world champion at the game of Go was built using RL.