Practice (Python+OpenCV)

Yu-Cheng, Wu

2019/07/15

Python3

- Python3.5+
- Array operation:
  - NumPy → pip3 install numpy
- Image processing:
  - OpenCV → pip3 install opencv-python
  - Pillow → pip3 install Pillow

Python Grammar

- Tutorial
  - http://cs231n.github.io/python-numpy-tutorial/
- Google it

NumPy

- NumPy is the fundamental package for scientific computing with Python. It provides containers and efficient tools to deal with multi-dimensional arrays.
- Tutorial
  - http://cs231n.github.io/python-numpy-tutorial/

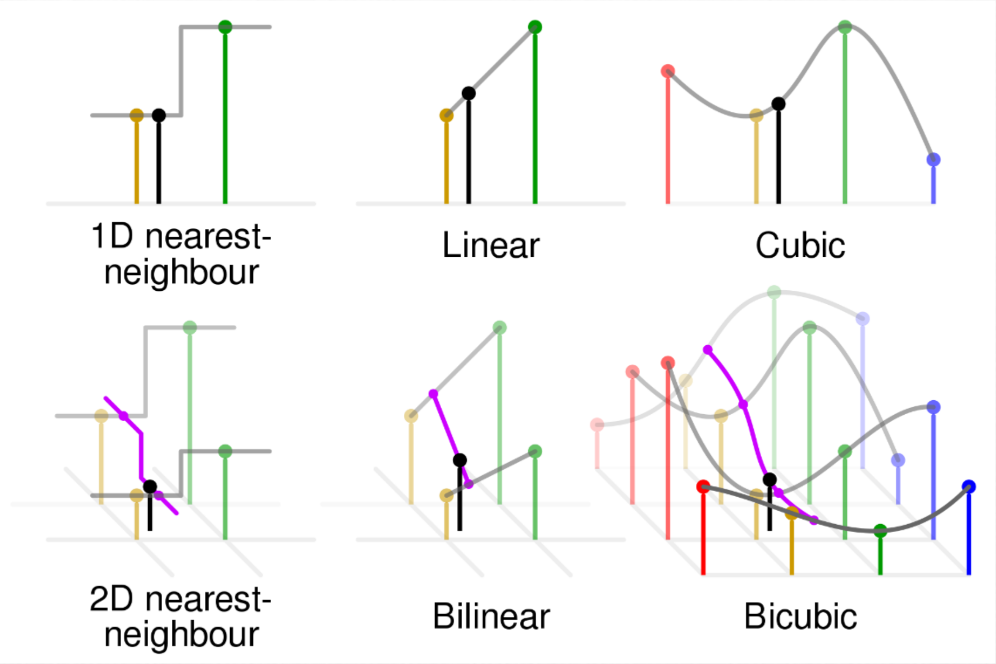
CV in Python

- OpenCV:
  - Popular library for computer vision application in C++ and python.
  - https://opencv-python-tutroals.readthedocs.io/en/latest/
- Pillow
  - PIL (Python Image Library) fork
  - https://pillow.readthedocs.io/en/stable/

Lab1

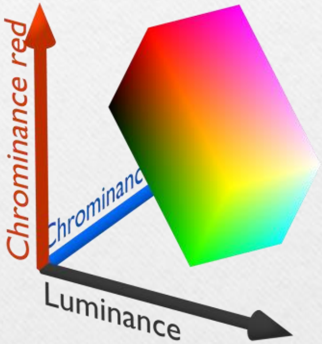
- Image operation
- Image smoothing
- Image denoising
- Image PCA analysis

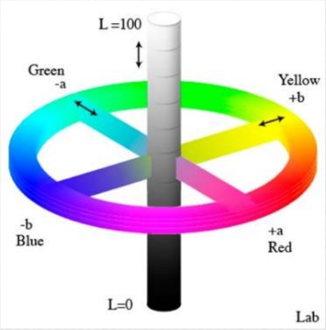
Interpolation



Color Space

- Organization of colors
  - RGB
  - YCbCr (Y: luminance, Cb, Cr: chrominance)
  - LAB
  - HSV (hue, saturation, value)





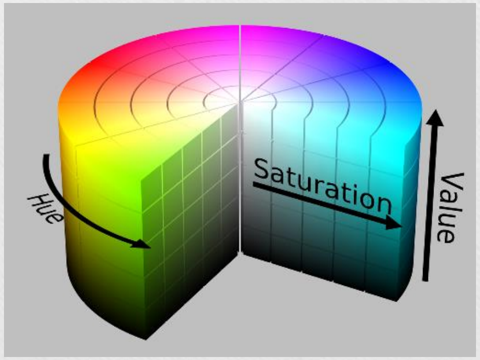


Image Filtering

- Convolution

$$g(x, y) = \frac{1}{W} \sum_{i,j \in [-r,r]} h(i, j) f(x - i, y - j) \qquad W = \sum_{i,j \in [-r,r]} h(i, j)$$

$$\begin{bmatrix} 45 & 60 & 98 & 127 & 132 & 133 & 137 & 133 \\ 46 & 65 & 98 & 123 & 126 & 128 & 131 & 133 \\ 47 & 65 & 96 & 115 & 119 & 123 & 135 & 137 \\ 47 & 63 & 91 & 107 & 113 & 122 & 138 & 134 \\ 50 & 59 & 80 & 97 & 110 & 123 & 133 & 134 \\ 49 & 53 & 68 & 83 & 97 & 113 & 128 & 133 \\ 50 & 50 & 58 & 70 & 84 & 102 & 116 & 126 \\ 50 & 50 & 52 & 58 & 69 & 86 & 101 & 120 \end{bmatrix} * \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} = \begin{bmatrix} 69 & 95 & 116 & 125 & 129 & 132 \\ 68 & 92 & 110 & 120 & 126 & 132 \\ 66 & 86 & 104 & 114 & 124 & 132 \\ 62 & 78 & 94 & 108 & 120 & 129 \\ 57 & 69 & 83 & 98 & 112 & 124 \\ 53 & 60 & 71 & 85 & 100 & 114 \end{bmatrix}$$

$$f(x,y) \qquad\qquad h(x,y) \qquad\qquad g(x,y)$$

Image Filtering

- Box filtering (average filtering)

$$h(i,j) = \frac{1}{r^2}$$

$$g(x,y) = \frac{1}{r^2} \sum_{i,j \in [-r,r]} f(x-i, y-j)$$

Image Filtering

- Gaussian filtering

$$h(i,j) = e^{-\frac{i^2+j^2}{2\sigma^2}}$$

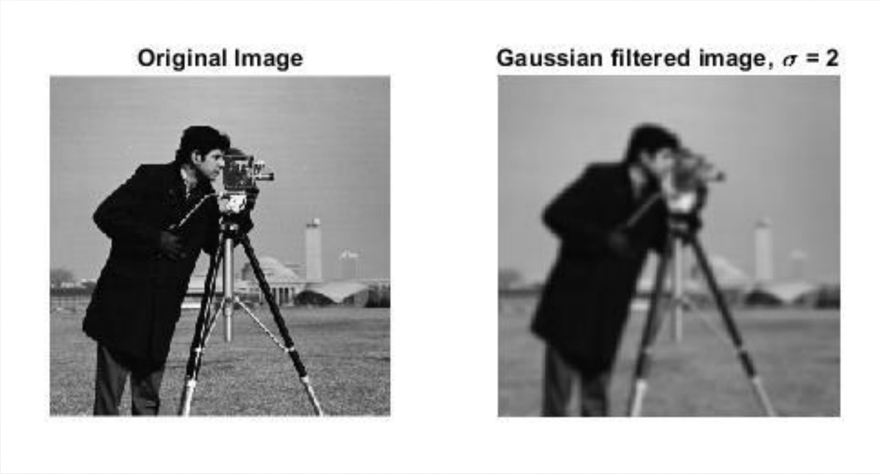
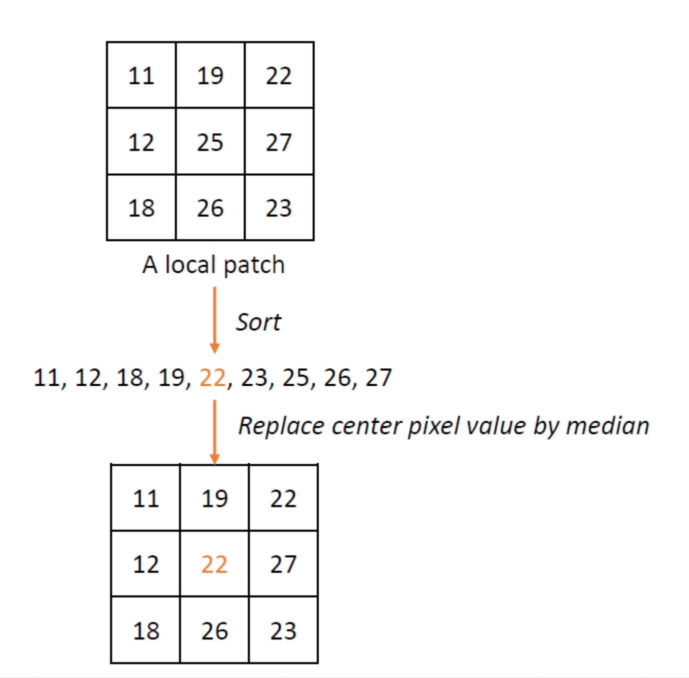


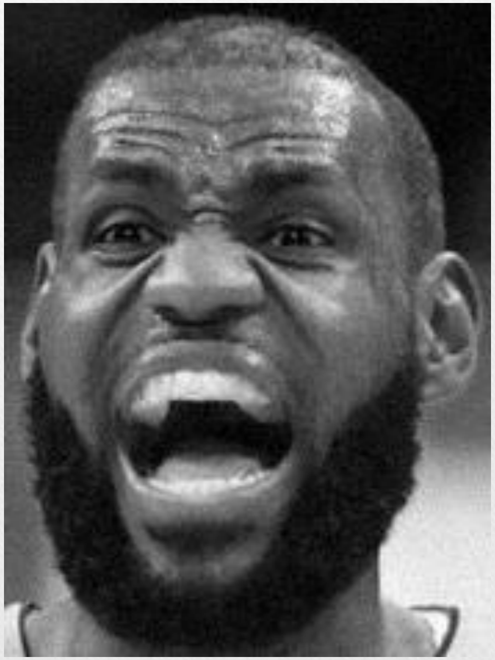
Image Filtering

- Median filtering
  - 3*3 example:



Noise

original



Gaussian noise

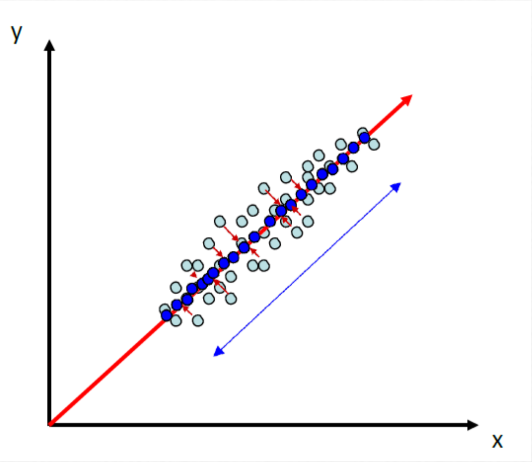
Salt and pepper noise

Lab1

- Image operation
  - Resize: down-sample, up-sample with different interpolation methods
  - Color conversion: RGB (BGR) to YCbCr
- Image smoothing
  - Averaging, Gaussian filtering, median filtering
- Image denoising
  - Add noise: Gaussian noise, salt and pepper noise
  - Denoise: Gaussian filtering, median filtering

Principal Component Analysis

- Unsupervised & linear dimension reduction
- Goal: determine the projection to maximize the variation of projected data



Formulation & Derivation for PCA

- Input: a set of instances ***x*** (N instances)
- Output: a projection vector ***w*** maximizing the variance of the projected data

$$\max_w E[(w^T x - w^T \mu)^2],\ \|w\|_2 = 1,\ \mu = \sum_{i=1}^{N} x_i$$

$$w \in \mathbb{R}^d,\ x_i \in \mathbb{R}^d,\ \forall i \in \{1, ..., N\}$$

Formulation & Derivation for PCA

- $$\begin{aligned} E[(w^T x - w^T \mu)^2] &= E[(w^T x - w^T \mu)(x^T w - \mu^T w)] \\ &= w^T E[(x-\mu)(x-\mu)^T] w \\ &= w^T \Sigma w \end{aligned}$$
- Covariance matrix

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} \{(x_i - \mu)(x_i - \mu)^T\}$$

Formulation & Derivation for PCA

- Cont'd

$$\max_{w} w^T \Sigma w, \ \ \|w\|_2 = 1$$

- Lagrangian multiplier

$$J(w) = w^T \Sigma w - \lambda(w^T w - 1)$$

$$\Rightarrow \frac{\partial J}{\partial w} = 2\Sigma w - 2\lambda w = 0$$

$$\Rightarrow w^T \Sigma w = \lambda w^T w = \lambda$$

Eigenanalysis & PCA

- Eigen decomposition

$$\Sigma W = W\Lambda \Rightarrow \Sigma = W\Lambda W^{-1} = W\Lambda W^T = \sum_{i=1}^{d} \lambda_i w_i w_i^T$$

- Next principle component

$$\begin{aligned} E[(w^T(1 - w_1 w_1^T)(x-\mu))^2] &= w^T(1 - w_1 w_1^T)\Sigma(1 - w_1 w_1^T)w \\ &= w^T(\Sigma - w_1 w_1^T \Sigma - \Sigma w_1 w_1^T + w_1 w_1^T \Sigma w_1 w_1^T)w \\ &= w^T(\Sigma - w_1 w_1^T \lambda_1 - \lambda_1 w_1 w_1^T + w_1 \lambda_1 w_1^T)w \\ &= w^T(\Sigma - \lambda_1 w_1 w_1^T)w \end{aligned}$$

Eigenanalysis & PCA

- Covariance matrix

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}\{(x_i-\mu)(x_i-\mu)^T\}$$

  - Rank = N-1 (if d > N-1)
    - → dim(Null space) = d-(N-1)
    - → at most N-1 non-zero eigenvalues

$$x_j - \mu = \sum_{i=1}^{d} a_i w_i$$
$$= \sum_{i=1}^{N-1} a_i w_i$$

Eigenanalysis & PCA

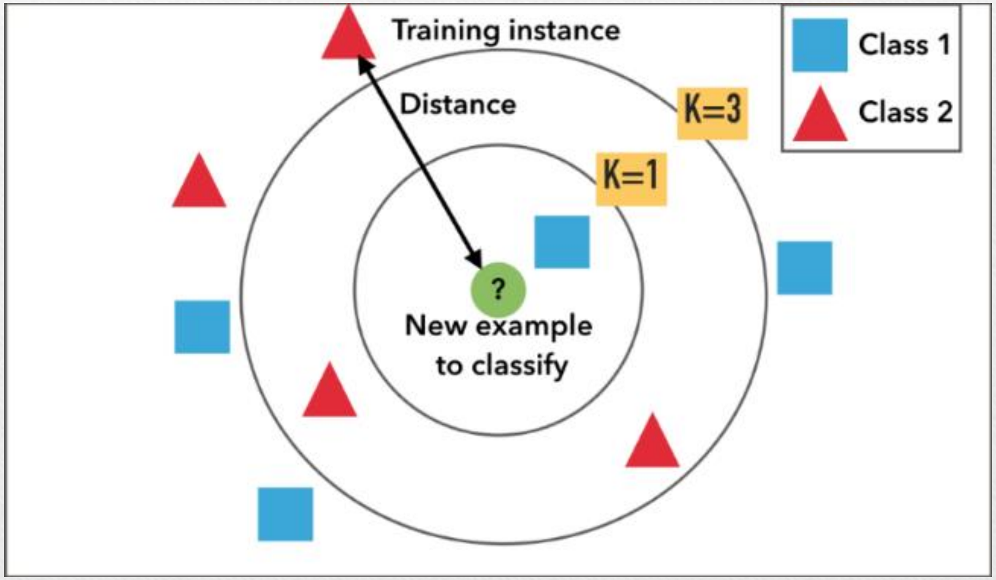
- Covariance matrix is symmetric
  - → eigenvectors are orthogonal
  - → span the instances with eigenvectors
- Less bases → larger reconstruction error

$$
\begin{aligned}
w_j^T(x_k - \mu) &= w_j^T \sum_{i=1}^{N-1} a_i w_i \\
&= \sum_{i=1}^{N-1} a_i \delta_{ij} \\
&= a_j
\end{aligned}
$$

$$
\begin{aligned}
x_j - \mu &= \sum_{i=1}^{N-1} a_i w_i \\
&\simeq \sum_{i=1}^{n} a_i w_i
\end{aligned}
$$

KNN Classifier

- k-nearest neighbors classifier



Lab1

- Image PCA analysis
  - 40 classes, 10 images for each class (6 train, 4 test)
  - N = 6*40, d = 56*46
  - Dim reduction: n = 3 or 100
  - KNN classifier: k = 1 or 3

$$x_i \in \mathbb{R}^d$$

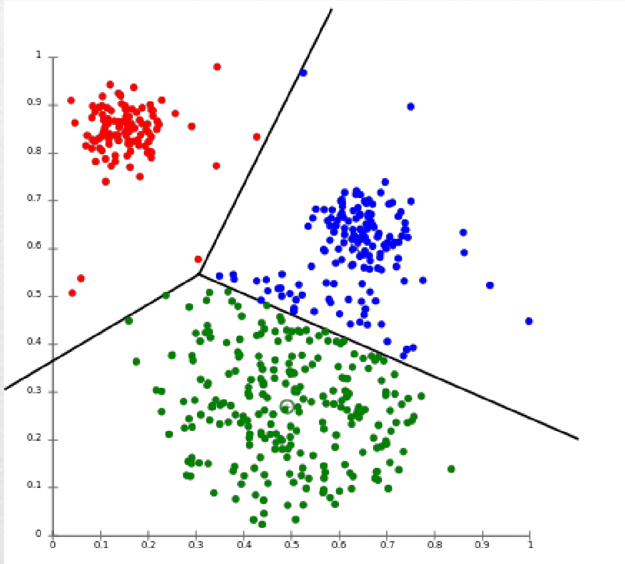
$$x_j - \mu = \sum_{i=1}^{N-1} a_i w_i \simeq \sum_{i=1}^{n} a_i w_i$$

Lab2

- Color segmentation
- Texture segmentation
- Feature descriptor
- Recognition with bag of visual words

K-means Clustering

- Group the data into k groups and minimize the sum of distances between data and corresponded group center (mean of each group)



K-means Clustering

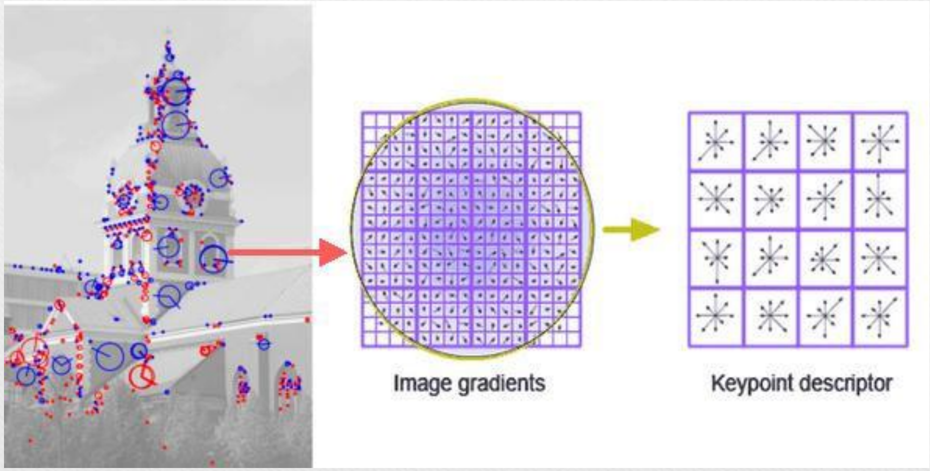
- Hard coding dictionary learning
  - *D*: group centers, *α*: belong to which group

$$\min_{D,\{\alpha_i\}} \sum_{i=1}^{N} \|x_i - D\alpha_i\|_2,\ s.t.\ \|\alpha_i\|_0 = 1,\ \|\alpha_i\|_1 = 1,\ \forall i \in \{1, ..., N\}$$

$$D \in \mathbb{R}^{d \times k},\ x_i \in \mathbb{R}^d,\ \alpha_i \in \mathbb{R}^k,\ \forall i \in \{1, ..., N\}$$

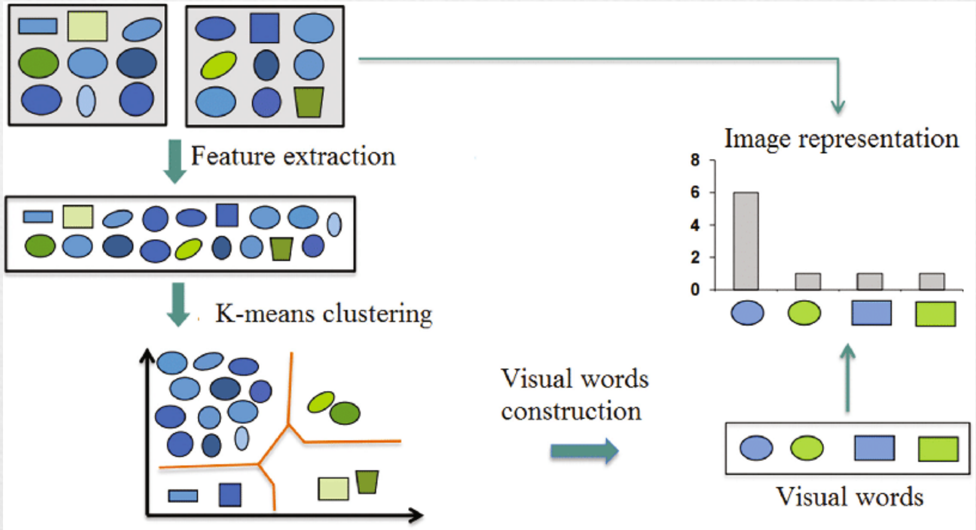
Feature Descriptor

- Feature extraction: feature detection + feature description
- Common descriptors: SURF, SIFT, BRIEF, ORB



Bag of Visual Words

- Encode images with visual words
- Can be used in image classification



Lab2

- Color segmentation
  - K-means clustering: d = 3, N = H*W, k = 10
- Texture segmentation
  - K-means clustering: d = 38 or 41, N = H*W, k = 6
- Bag of visual words
  - 5 classes, 100 train and 100 test images for each class
  - K-means clustering: d = 128 (dim of SURF feat.), N = sum(# of SURF feat. in each image), k = 50
  - BoW feat: dim = 50