Python OpenCV Lab

Yu-Cheng Wu

2020/07/20

Outline

- Prerequisite
- Lab1: basic image processing
- Lab2: homography

Prerequisite

- Code editor: VS Code, Notepad++
- Language: Python3
- Library:
  - NumPy: array operation
  - OpenCV: computer vision task
  - Matplotlib: visualization in python

Python Grammar

- Tutorial: http://cs231n.github.io/python-numpy-tutorial/, Google
- Basic
  - print(), if else, for loop
- Useful
  - List, indexing of list
    - Ex: a = [1,2,4]; a[0] = 1; a[-1] = 4; a[:2] = [1,2]
- Optional
  - Function declaration (if some process is repeated)

NumPy

- Tutorial: http://cs231n.github.io/python-numpy-tutorial/, Google
- Import library:
  - import numpy as np
- Basic:
  - Array initialization, basic property (shape, data type), indexing
- Useful:
  - Build-in function for array operation: argmin, matmul

OpenCV

- Tutorial: https://opencv-python-tutroals.readthedocs.io/en/latest/, Google
- Basic
  - Image read, write, resize, color conversion, …
- Useful
  - Padding, filtering, other CV tasks

Lab1: Basic Image Processing

- Image operation
- Image filtering
- Image PCA analysis

Image Filtering

- Weighted sum of the region of the input

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

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

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

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

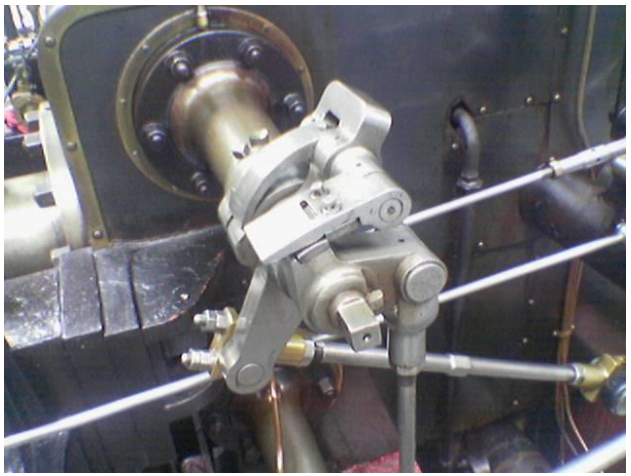
$g(x,y)$

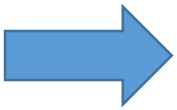
Lab1: Image Filtering

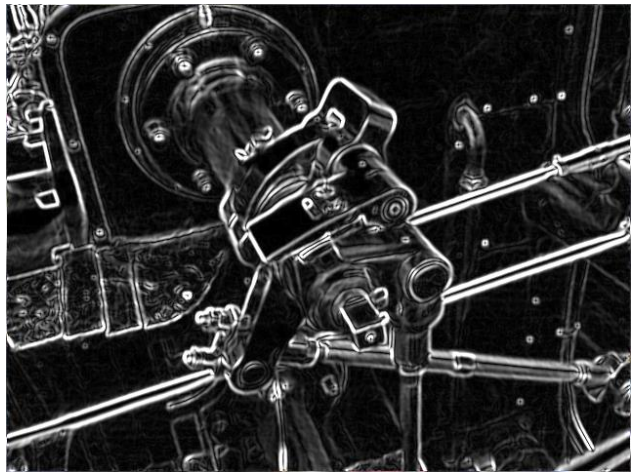
- Sobel filter: used in edge detection

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

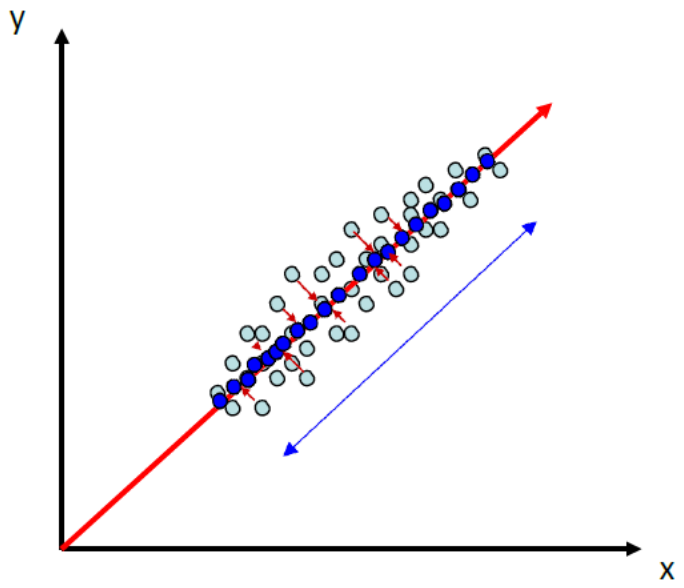






Principal Component Analysis (PCA)

- Goal: determine the projection to maximize the variance of the projected data
- Linear dimension reduction



PCA

- Input:
  - A set of instances $\{\vec{x}_i\}_{i=1}^N, \vec{x}_i \in \mathbb{R}^d$
  - Zero mean: $\vec{x'}_i = \vec{x}_i - \vec{\mu}$, where $\vec{\mu} = \frac{1}{N}\sum_i \vec{x}_i$
- First component:
  - A unit vector $\vec{w} \in \mathbb{R}^d$ that maximize the variance of the projected data $\{\vec{w} \cdot \vec{x'}_i\}_{i=1}^N$
- Further components:
  - Derived from the data without the first component

$$\{\vec{x'}_i\}_{i=1}^N \rightarrow \{\vec{x}_i - (\vec{w} \cdot \vec{x'}_i)\vec{w}\}_{i=1}^N$$

  - Mutually orthogonal

Dimension Reduction with PCA

- Principal components (PCA eigenbasis) $\{\vec{w}_i\}_{i=1}^{K}$ (usually N < d → K = N-1)
- Vector representation

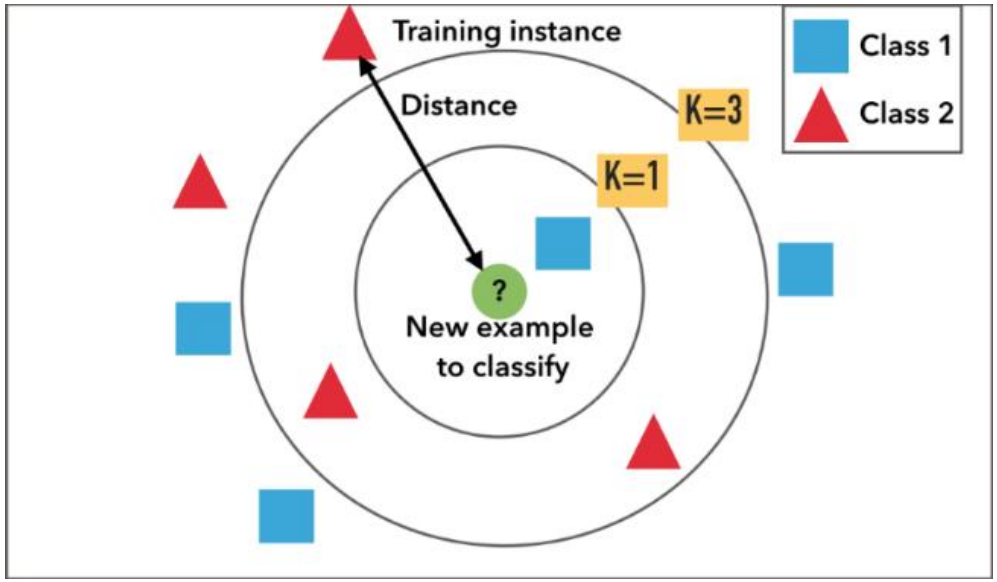
$$\vec{x}_i = \vec{\mu} + \sum_{i=1}^{N-1}(\vec{w}_i \cdot (\vec{x}_i - \vec{\mu}))\vec{w}_i$$

- Vector approximation

$$\vec{x}_i \cong \vec{\mu} + \sum_{i=1}^{k}(\vec{w}_i \cdot (\vec{x}_i - \vec{\mu}))\vec{w}_i$$

KNN Classifier

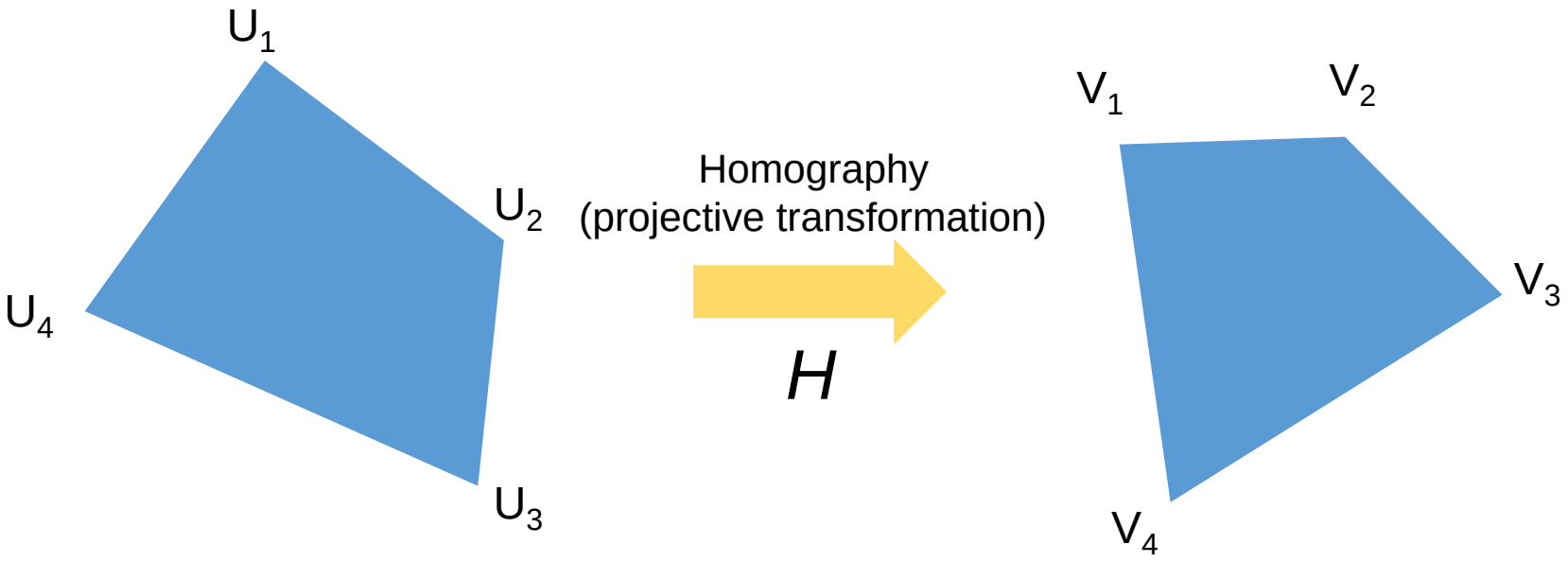
- k-nearest neighbors classifier



Lab1: Image PCA Analysis

- Given face images $\vec{x}_i$ with 40 classes, 10 images for each class (6 train, 4 test)
  - Perform PCA on training set → get the eigenfaces $\vec{w}_i$
  - Reconstructed an image with 3 or 100 eigenfaces and compute mean square error (MSE)
  - Apply kNN classifer on testing set

Lab2: Homography



Recap of Homography

- Matrix form:

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

- Equations

$$v_x = \frac{h_{11}u_x + h_{12}u_y + h_{13}}{h_{31}u_x + h_{32}u_y + h_{33}}$$

$$v_y = \frac{h_{21}u_x + h_{22}u_y + h_{23}}{h_{31}u_x + h_{32}u_y + h_{33}}$$

Recap of Homography

- Degree of freedom:
  - 9 - 1 = 8 DoF

$$v_x = \frac{kh_{11}u_x + kh_{12}u_y + kh_{13}}{kh_{31}u_x + kh_{32}u_y + kh_{33}} \qquad v_y = \frac{kh_{21}u_x + kh_{22}u_y + kh_{23}}{kh_{31}u_x + kh_{32}u_y + kh_{33}}$$

$$v_x = \frac{h_{11}u_x + h_{12}u_y + h_{13}}{h_{31}u_x + h_{32}u_y + h_{33}} \qquad v_y = \frac{h_{21}u_x + h_{22}u_y + h_{23}}{h_{31}u_x + h_{32}u_y + h_{33}}$$

- Solution: solve h_{ij} with constraint

$$h_{11}^2 + \ldots + h_{33}^2 = 1$$

Solution

$$v_x = \frac{h_{11}u_x + h_{12}u_y + h_{13}}{h_{31}u_x + h_{32}u_y + h_{33}}$$

$$v_y = \frac{h_{21}u_x + h_{22}u_y + h_{23}}{h_{31}u_x + h_{32}u_y + h_{33}}$$

subjective to $h_{11}^2 + ... + h_{33}^2 = 1$

$$(h_{31}u_x + h_{32}u_y + h_{33})v_x = h_{11}u_x + h_{12}u_y + h_{13}$$

$$(h_{31}u_x + h_{32}u_y + h_{33})v_y = h_{21}u_x + h_{22}u_y + h_{23}$$

$$h_{11}u_x + h_{12}u_y + h_{13} - h_{31}u_xv_x - h_{32}u_yv_x - h_{33}v_x = 0$$

$$h_{21}u_x + h_{22}u_y + h_{23} - h_{31}u_xv_y - h_{32}u_yv_y - h_{33}v_y = 0$$

Solution

- Construct a linear system using N vertices:

$$\underset{2N \times 9}{\mathbf{A}} \quad \underset{9 \times 1}{\mathbf{h}} = \underset{2N \times 1}{\mathbf{b}}$$

  - **b** is all zero
  - Solve **h**:
    - **Ah** = 0
    - $\mathbf{A}^T\mathbf{A}\mathbf{h} = 0$
    - SVD of $\mathbf{A}^T\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$
    - Let **h** be the column of **U** (unit eigenvector) associated with the smallest eigenvalue in **Σ**

Lab2 Problem

Make the QR code frontal parallel

Backward Warping

- Prevent holes in output space
- Pixel value at sub-pixel location like (30.21, 22.74)?
  - Bilinear interpolation
  - Nearest neighbor