Python OpenCV Lab

2021 Crash Course

NTU GIEE
Media IC & System Lab
楊凱翔

2021/08/02

Outline

- Prerequisite
- Lab1: basic image processing
  - Image Filtering
  - Image PCA Analysis
- Lab2: Homography

Outline

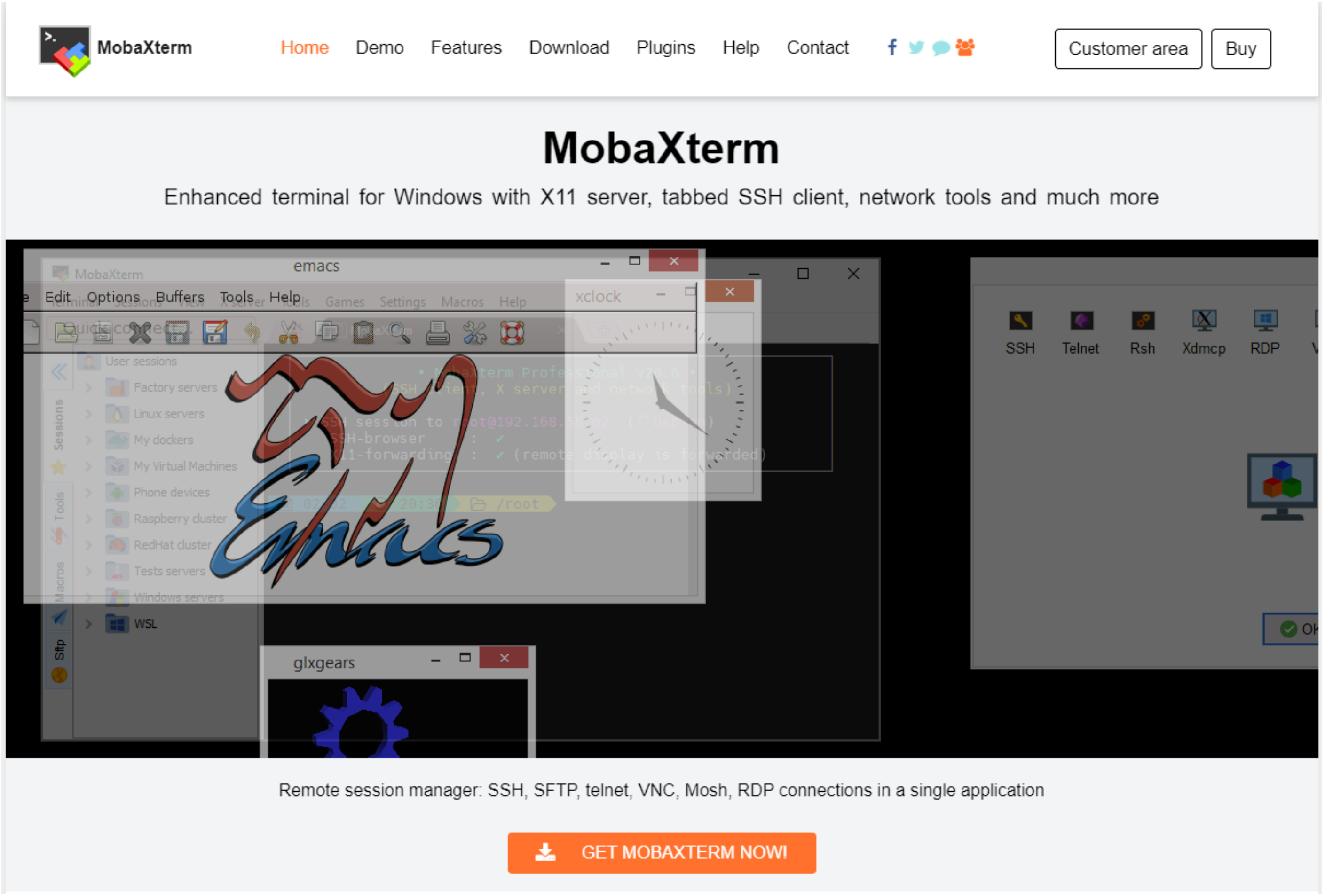
- Prerequisite
- Lab1: basic image processing
  - Image Filtering
  - Image PCA Analysis
- Lab2: Homography

Prerequisite

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

MobaXterm

- Download from https://mobaxterm.mobatek.net/

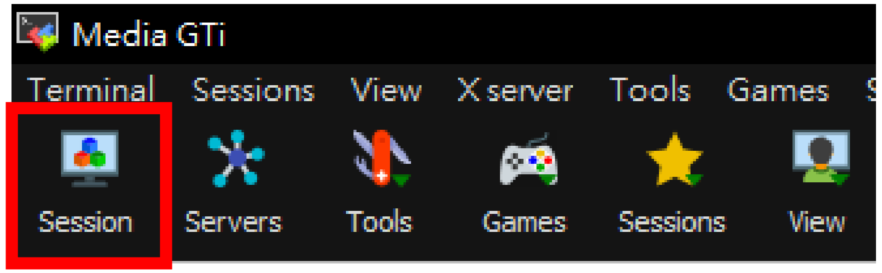


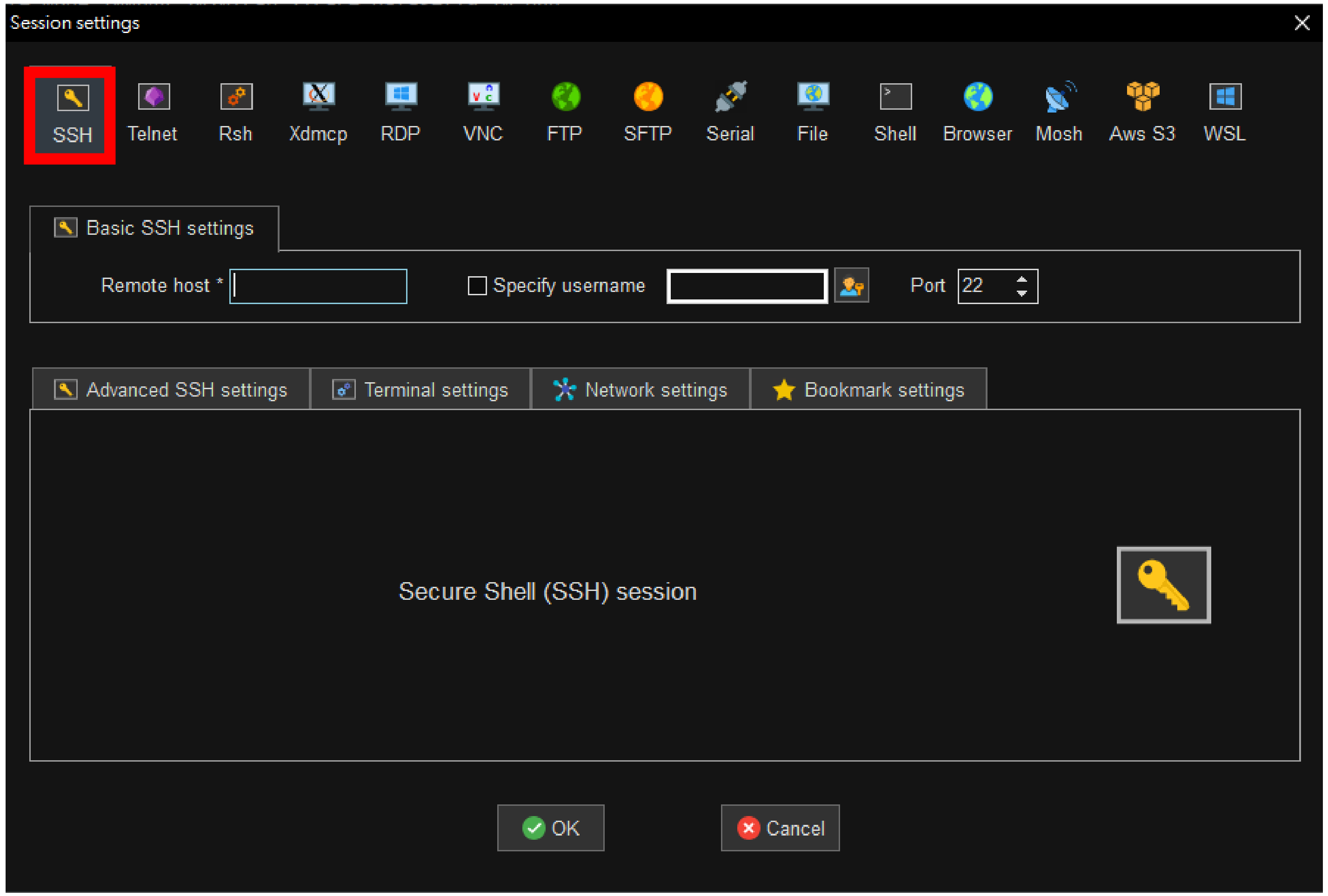
MobaXterm

Enhanced terminal for Windows with X11 server, tabbed SSH client, network tools and much more

Remote session manager: SSH, SFTP, telnet, VNC, Mosh, RDP connections in a single application

MobaXterm





Remote host *

Specify username

Port 22

Refer to 421 wiki to get remote host and port.

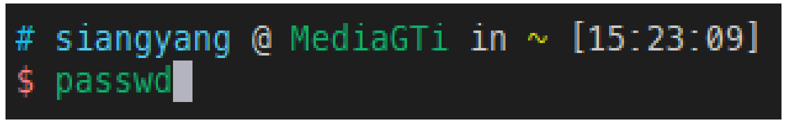
MediaGTi :

remote host: 140.112.48.127

port: 10800

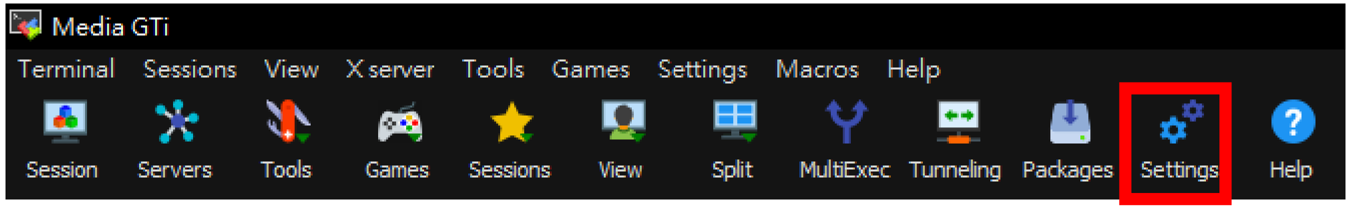
MobaXterm

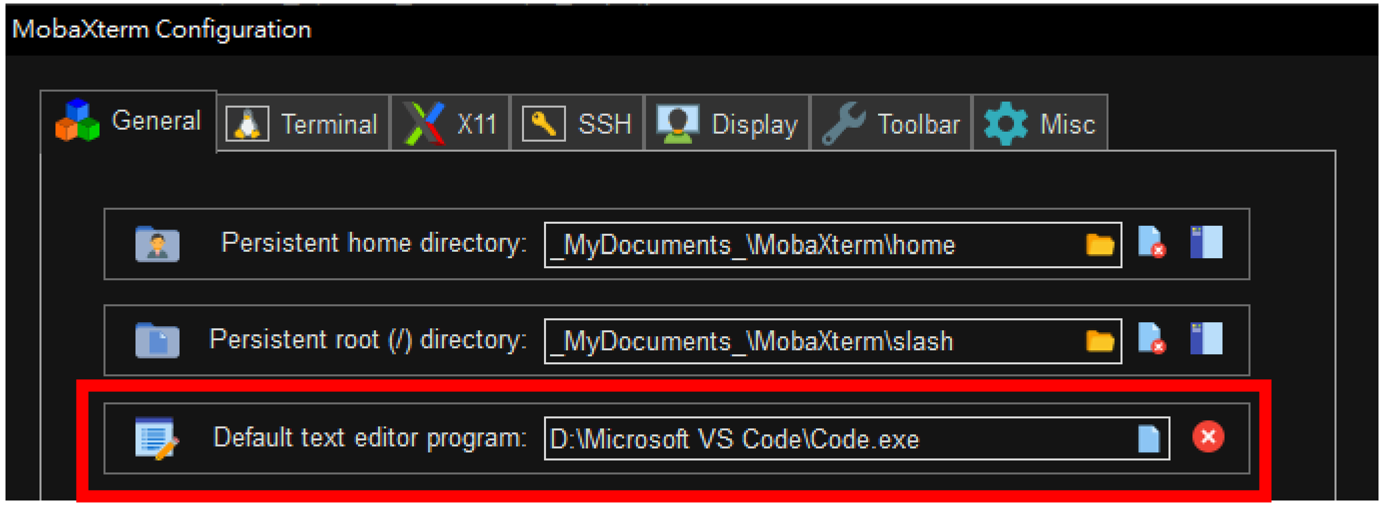
- Type `$ passwd` to change password



```
# siangyang @ MediaGTi in ~ [15:23:09]
$ passwd
```

- Change default text editor


Media GTi
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help



Persistent home directory: _MyDocuments_\MobaXterm\home

Persistent root (/) directory: _MyDocuments_\MobaXterm\slash

Default text editor program: D:\Microsoft VS Code\Code.exe

Linux Commend

- Tutorial: https://blog.techbridge.cc/2017/12/23/linux-commnd-line-tutorial/, Google
- Basic
  - cd, ls, git, mkdir, mv, rm, …
- Useful tool
  - tmux, nvidia-htop, nvidia-smi, jupyter notebook, anaconda, …

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://docs.opencv.org/4.5.2/, Google
- Basic
  - Image read, write, resize, color conversion, …
- Useful
  - Padding, filtering, other CV tasks

[Supplement] some tips for image read/write

- All image read method
  - OpenCV
  - Pillow
  - Matplotlib
  - Scikit-image
  - Imageio
  - Scipy.misc (unavailable after 1.1.0)
- Shape: (H, W, C)
- In OpenCV, the default color order is BGR
- Change to np.array and change dtype from uint8 to floatXX before some operations
- More examples in ImageRead.ipynb

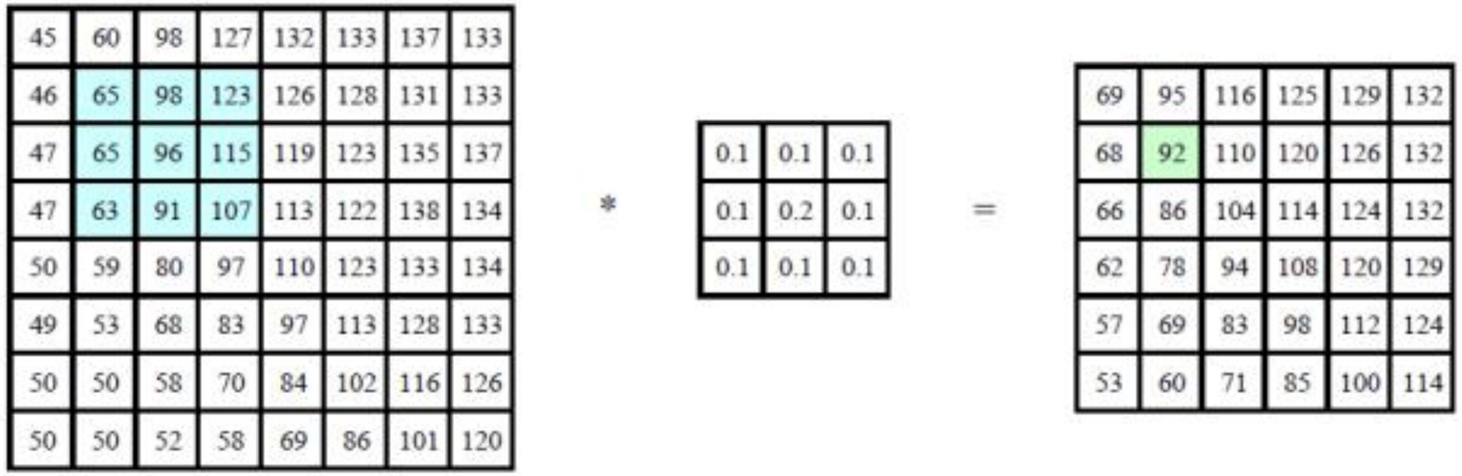
Outline

- Prerequisite
- Lab1: basic image processing
  - Image Filtering
  - Image PCA Analysis
- Lab2: Homography

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

$g(x,y)$

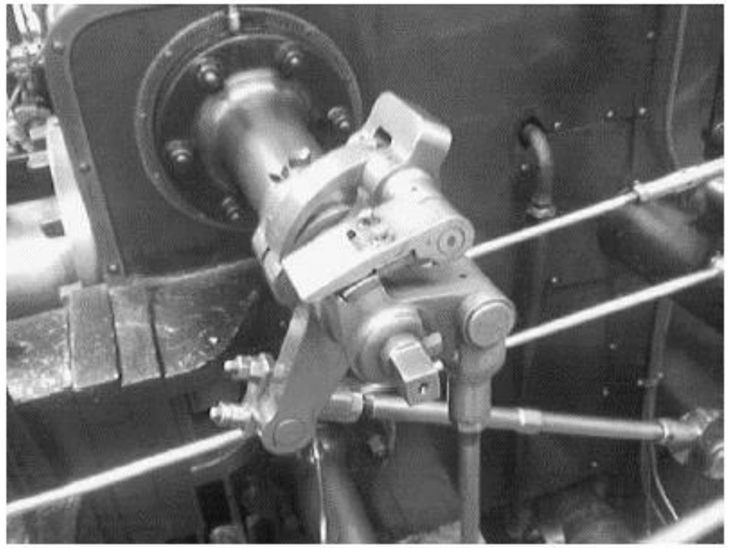
Lab1: Image Filtering

- Sobel filter: used in edge detection

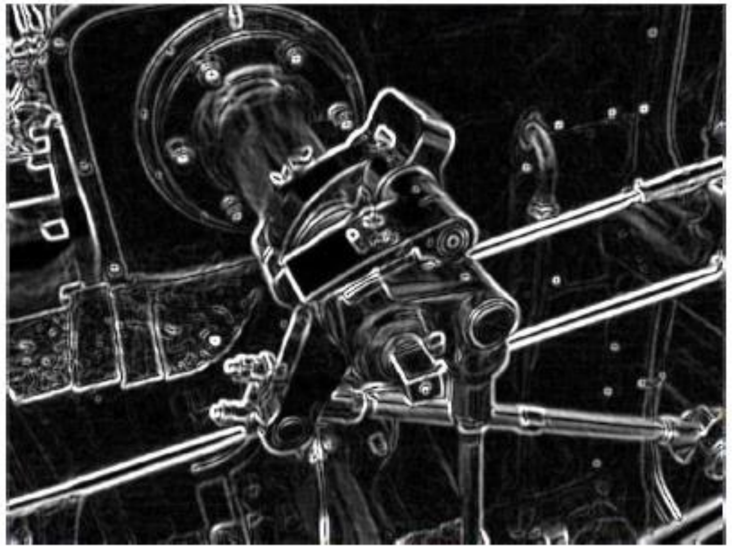
$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

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

Input Gray-scale image

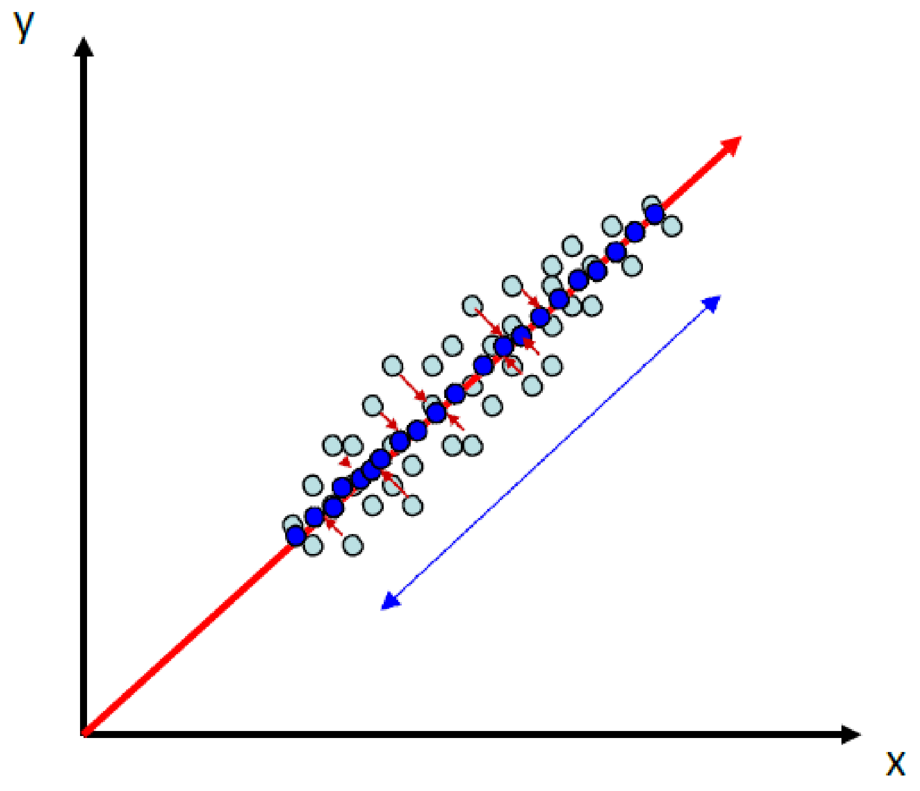


Output Gray-scale image



Principal Component Analysis (PCA)

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



Principal Component Analysis (PCA)

- Input:
  - A set of instances $\{\vec{x}\}_{i=1}^{N}, \vec{x}_i \in \mathbb{R}^d$
  - Zero mean: $\vec{x}' = \vec{x} - \vec{\mu}$, where $\vec{\mu} = \frac{1}{N}\sum_i \overrightarrow{x_i}$
- 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

Principal Component Analysis (PCA)

$$var(\omega^T x) \rightarrow E\{(\omega^T x - \omega^T \mu) \cdot (\omega^T x - \omega^T \mu)^T\} = \omega^T \underbrace{E\{(x-\mu) \cdot (x-\mu)^T\}}_{\Sigma} \omega$$

We need to maximize $var(\omega^T x)$ with $\|\omega\| = \omega^T \omega = 1$
$\rightarrow max(\omega^T \Sigma \omega)$

$$\mathrm{L}(\omega) = max(\omega^T \Sigma \omega) - \lambda(\omega^T \omega - 1)$$

$$\frac{\partial \mathrm{L}(\omega)}{\partial \omega} = 2 \cdot \Sigma \cdot \omega - 2 \cdot \lambda \cdot \omega = 0 \ (peek\ value)$$

$$\Sigma \cdot \omega - \lambda \cdot \omega = 0 \rightarrow \Sigma \cdot \omega = \lambda \cdot \omega \rightarrow \boxed{\omega^T \cdot \Sigma \cdot \omega = \lambda \cdot \omega^T \omega}$$

$solve\ SVD(\Sigma)$

Principal Component Analysis (PCA)

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

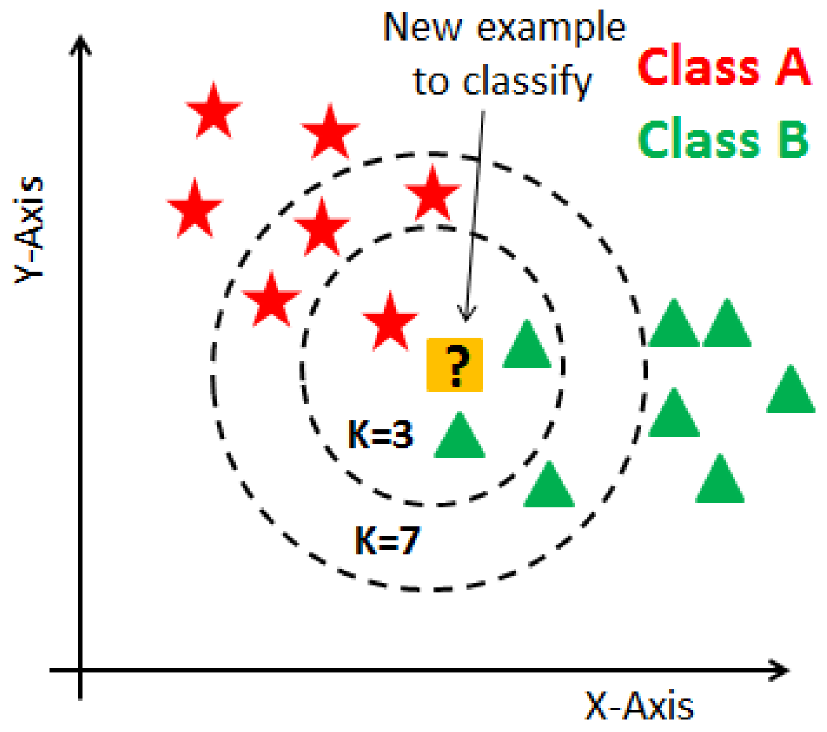
$$\vec{x}_i = \vec{\mu} + \sum_{i=1}^{N-1} (\vec{w}_i \cdot (\vec{x}_i - \vec{\mu}))\vec{w}_i$$

- 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

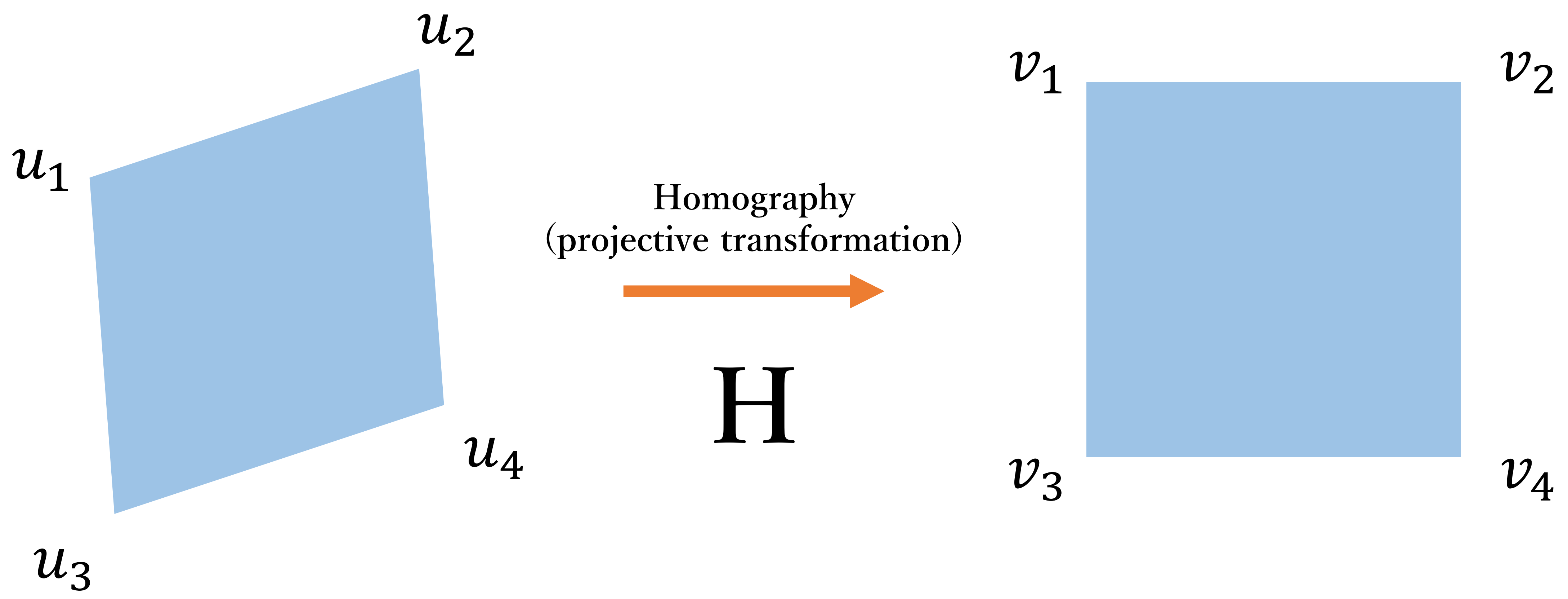


Lab1: Image PCA Analysis

- 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 classifier on test set

Outline

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

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

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

$$\Longrightarrow$$

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

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

- Constraint

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

Solution

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

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

$$\Rightarrow \quad \begin{aligned} (h_{31}u_x + h_{32}u_y + h_{33})v_x &= h_{11}u_x + h_{12}u_y + h_{13} \\ (h_{31}u_x + h_{32}u_y + h_{33})v_y &= h_{21}u_x + h_{22}u_y + h_{23} \end{aligned}$$

$$\Rightarrow \quad \begin{aligned} h_{11}u_x + h_{12}u_y + h_{13} - h_{31}u_xv_x - h_{32}u_yv_x - h_{33}v_x &= 0 \\ h_{21}u_x + h_{22}u_y + h_{23} - h_{31}u_xv_y - h_{32}u_yv_y - h_{33}v_y &= 0 \end{aligned}$$

Solution

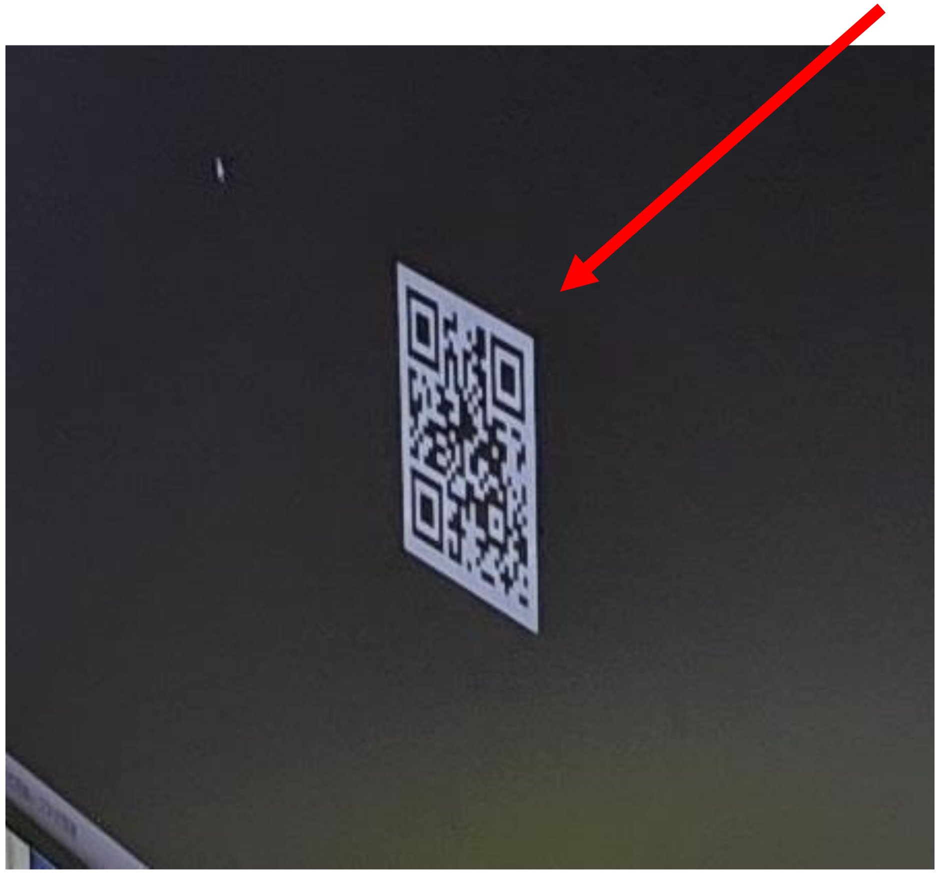
- Construct a linear system using N vertices:

$$\begin{array}{cccc} 2N \times 9 & 9 \times 1 & & 2N \times 1 \\ \mathbf{A} & \mathbf{h} & = & \mathbf{b} \end{array}$$

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

Lab2 Problem

Make the QR code frontal parallel



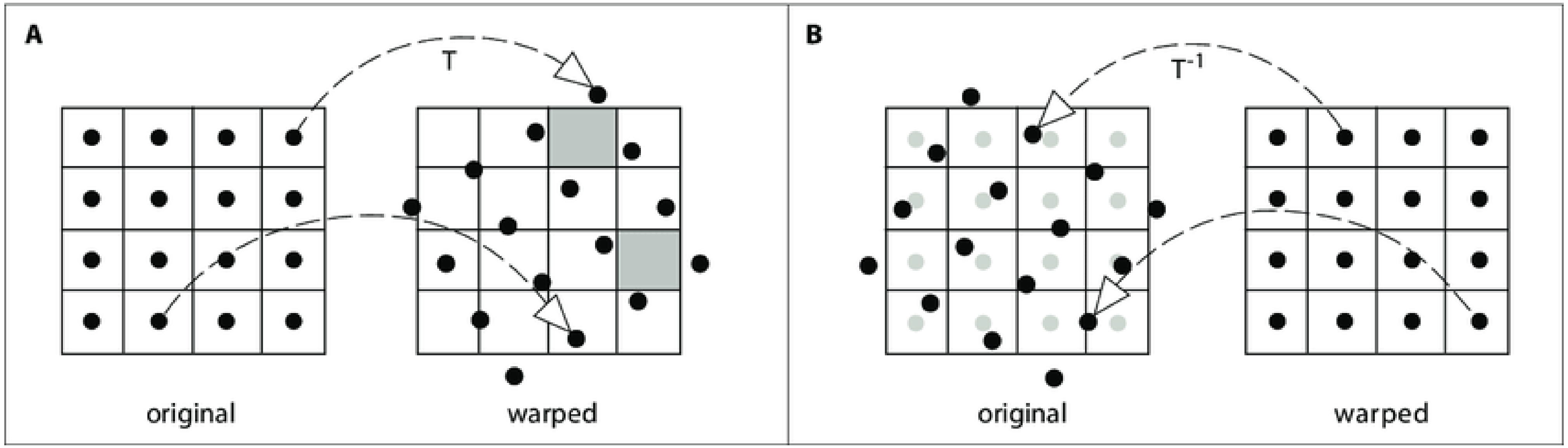
Calling function cv2.findHomography is FORBIDDEN!!!

Backward Warping

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

Forward Warping

Backward Warping


A
T
original
warped
B
T^{-1}
original
warped