南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

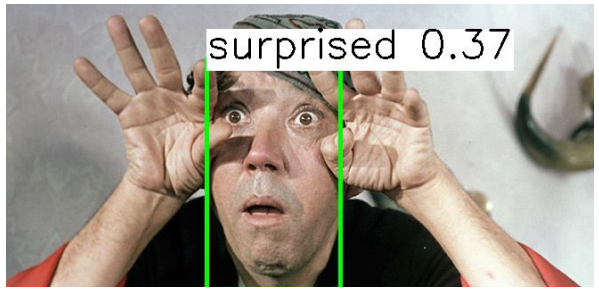
OpenCV and Deep Learning

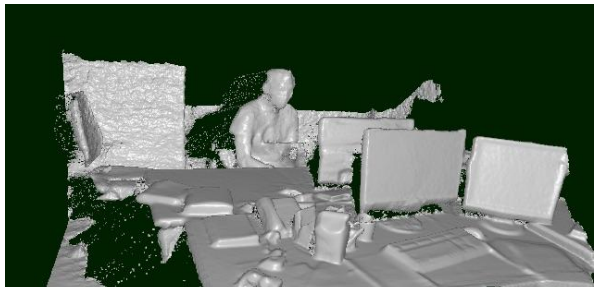
Wanli Zhong

2025-01-13

What Is OpenCV

- **Open**-source **C**omputer **V**ision Library (2000-present)
- 2500+ optimized computer vision and machine learning algorithms
- Implemented in C/C++, bindings for Python, Java, Javascript, Swift, Julia
- Free for research and commercial use (Apache 2 license)
- **80K+ stars** in GitHub: https://github.com/opencv/opencv
- Actively developed: latest version 4.11.0 released on Jan 10, 2025






opencv + opencv_contrib > 70 modules

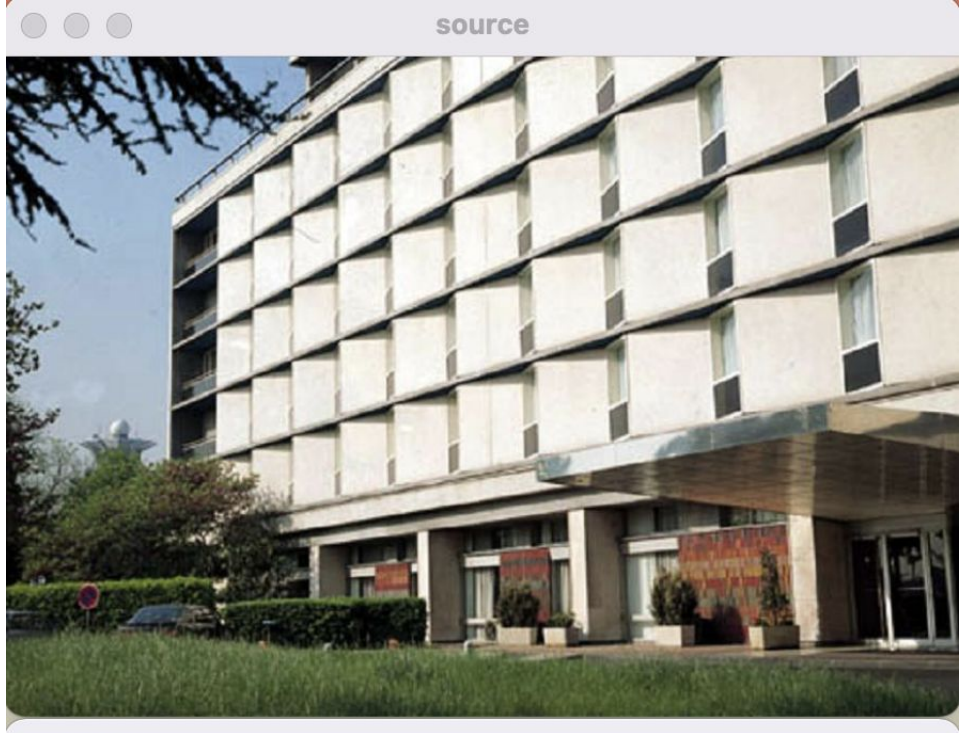
- **core**: Core Functionality
- **imgproc**: Image Processing
- **imgcodecs**: Image file reading and writing
- **videoio**: Video I/O
- **highgui**: High-level GUI
- **video**: Video Analysis
- **calib3d**: Camera Calibration and 3D Reconstruction
- **features2d**: 2D Features Framework
- **objdetect**: Object Detection
- **ml**: Machine Learning
- **flann**: Clustering and Search in Multi-Dimensional Spaces
- **photo**: Computational Photography
- **stitching**: Images stitching
- **gapi**: Graph API
- **dnn**: Deep Neural Network module

opencv + **opencv_contrib** > 70 modules

- **aruco**: Aruco markers, module functionality was moved to objdetect module
- **ccalib**: Custom Calibration Pattern for 3D reconstruction
- **cudastereo**: Stereo Correspondence
- **dnn_objdetect**: DNN used for object detection
- **dnn_superres**: DNN used for super resolution
- **rgbd**: RGB-Depth Processing
- **sfm**: Structure From Motion
- **shape**: Shape Distance and Matching
- **stereo**: Stereo Correspondance Algorithms
- **structured_light**: Structured Light API
- **surface_matching**: Surface Matching
- **viz**: 3D Visualizer
- **wechat_qrcode**: WeChat QR code detector for detecting and parsing QR code
- **……**
- **……**

Basic Usage:

```
import cv2 as cv

# load image
im = cv.imread('building.jpg')
# convert colour image to grey image
im_grey = cv.cvtColor(im, cv.COLOR_BGR2GRAY)
# Gaussian blur
im_blur = cv.GaussianBlur(im_grey, (3, 3), 0, 0)
# Canny edge detection
edges = cv.Canny(im_blur, 25, 75, 3)
# show results
cv.imshow('source', im)
cv.imshow('edges', edges)
cv.waitKey()
```



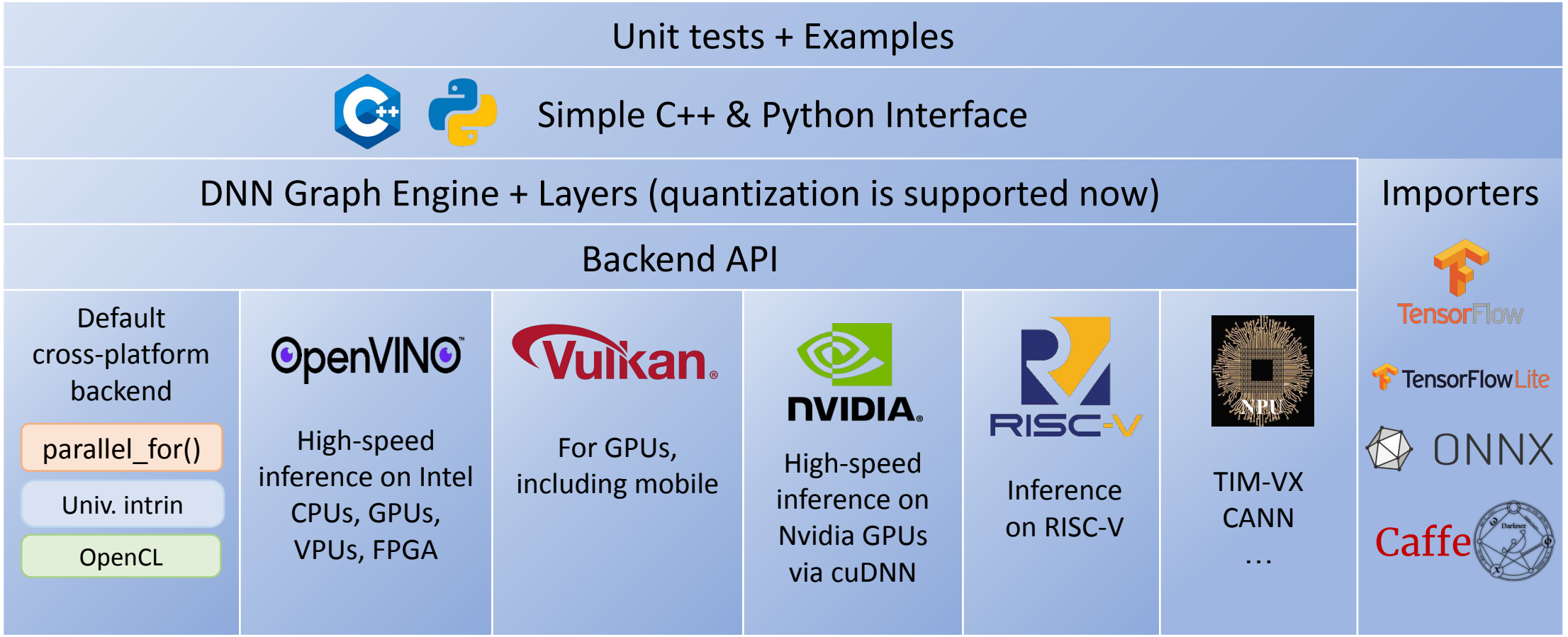
opencv + opencv_contrib > 70 modules

- **dnn: Deep Neural Network module.**

Inference framework, compatible with different network formats.

```
# 1. Load input and deep learning model
frame = cv.imread('lena.jpg')
net = cv.dnn.readNet('la_muse.t7')
# 2. Pre-processing
inWidth = frame.shape[1]
inHeight = frame.shape[0]
inp = cv.dnn.blobFromImage(frame, 1.0, (inWidth, inHeight),
                           (103.939, 116.779, 123.68), swapRB=False, crop=False)
# 3. Set model input
net.setInput(inp)
# 4. Infer
out = net.forward()
# 5. Post-processing
out = out.reshape(3, out.shape[2], out.shape[3])
out[0] += 103.939
out[1] += 116.779
out[2] += 123.68
out /= 255
out = out.transpose(1, 2, 0)
```

- Easy-to-use API
- Supports multiple devices (CPU, GPU, NPU, FPGA) and OS (Linux, Windows, MacOS, Android)
- Backend framework to leverage different acceleration library
- Users can implement custom layers in C++ or Python



General Usage

```
import cv2 as cv

# load the model
# automatically determine the format
net = cv.dnn.readNet(modelfile[, config[, framework]])

# configure the input
net.setInput(blob[, name[, scalefactor[, mean]]])

# infer
output = net.forward()

# postprocess and visualization
```

1. load deep learning model

2. set model input

3. execute inference

working with different devices and backends

```
import cv2 as cv

# load the model
# automatically determine the format
net = cv.dnn.readNet(modelfile[, config[, framework]])

net.setPreferableTarget()
net.setPreferableBackend()

# configure the input
net.setInput(blob[, name[, scalefactor[, mean]]])

# infer
output = net.forward()

# postprocess and visualization
```

1. load deep learning model

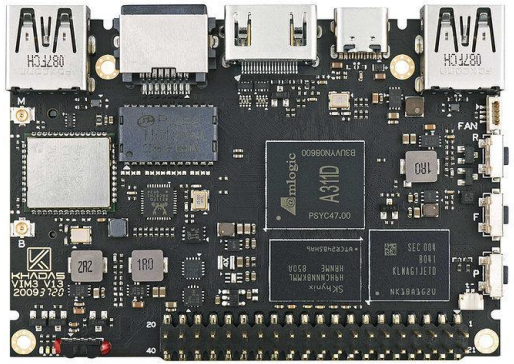
set device and backend

2. set model input

3. execute inference

Combination of Target&Backend

Target / Backend	CPU	OPENCL	OPENCL_FP16	MYRIAD	VULKAN	FPGA	CUDA CUDA_16	NPU
OPENCV	√	√	√	×	×	×	×	×
HALIDE	√	√	×	×	×	×	×	×
INFERENCE _ENGINE	√	√	√	√	×	√	×	×
VKCOM	×	×	×	×	√	×	×	×
CUDA	×	×	×	×	×	×	√	×
TIMVX	×	×	×	×	×	×	×	√
CANN	×	×	×	×	×	×	×	√

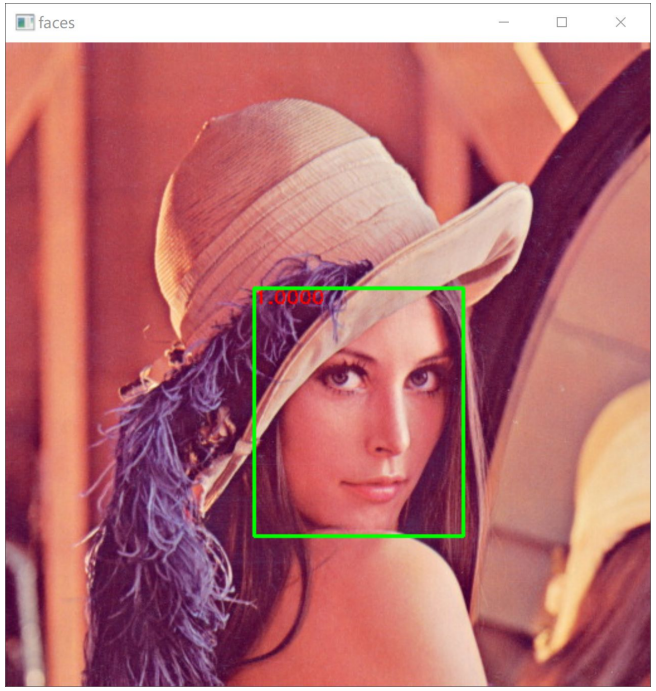


Khadas VIM3

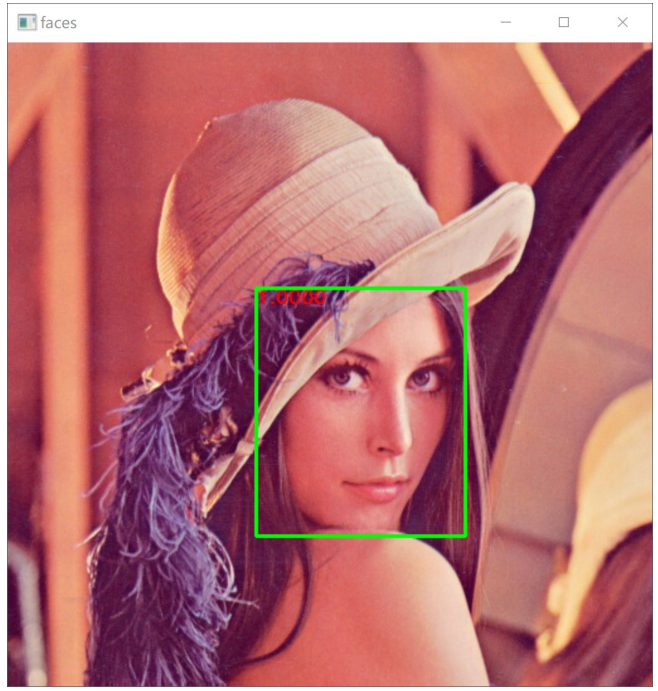
Specialized API

- text detection & recognition
  - cv.dnn.TextDetectionModel
  - cv.dnn.TextRecognitionModel
- face detection & recognition
  - cv.FaceDetectorYN
  - cv.FaceRecognizerSF
- object tracking
  - cv.TrackerDaSiamRPN
  - cv.TrackerNano
  - cv.TrackerVit
- bardcode & qrcode detection
  - cv.barcode.BarcodeDetector
  - cv.wechat_qrcode_WeChatQRCode

```
im = cv.imread('lena.jpg')
h, w, _ = im.shape

detector = cv.readNet('yunet.onnx')
inputBlob = cv.dnn.blobFromImage(image=im, size=[w,h])
detector.setInput(inputBlob)
outputBlob = net.forward(['loc', 'conf', 'iou'])
faces = postProcess(outputBlob)

im_res = visualize(im, faces)
cv.imshow('res', im_res)
cv.waitKey()
```



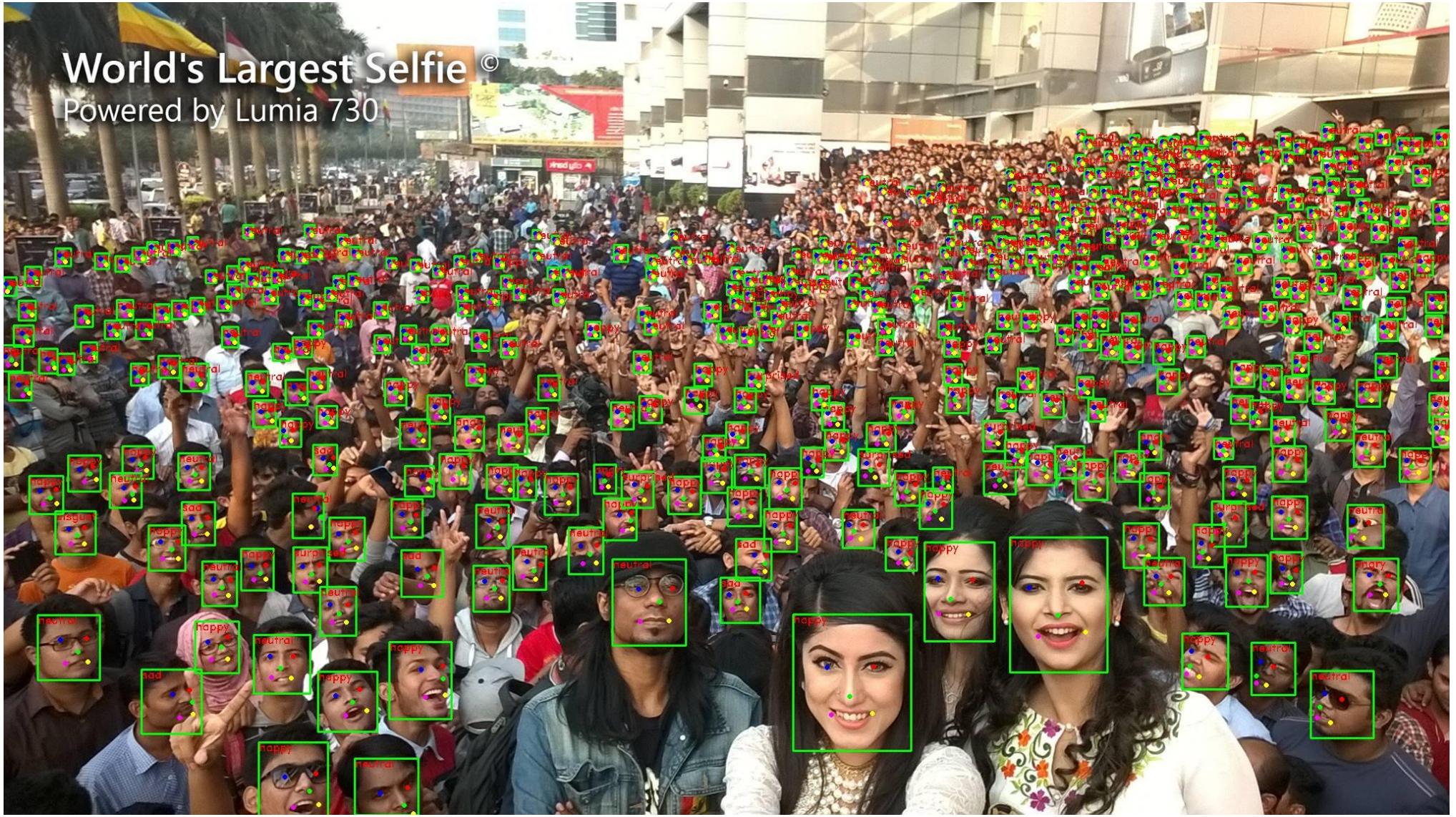
```
im = cv.imread('lena.jpg')
h, w, _ = im.shape

detector = cv.FaceDetectorYN.create(model='yunet.onnx
', input_size=[w,h], config='')
faces = detector.detect(im)



im_res = visualize(im, faces)
cv.imshow('res', im_res)
cv.waitKey()
```



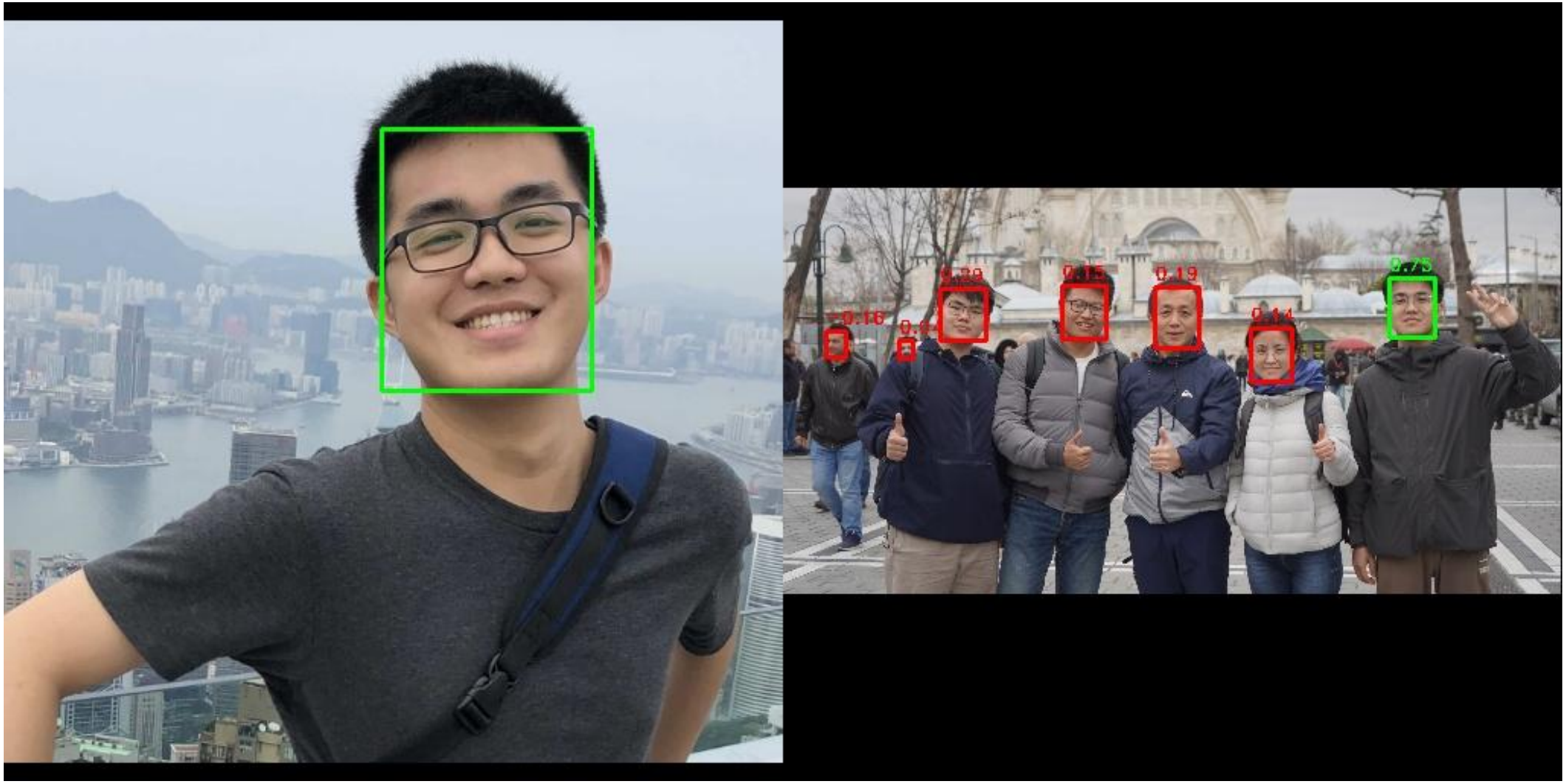
OpenCV Model Zoo & Benchmark

https://github.com/opencv/opencv_zoo

- A collection of high-quality free models
- ONNX format, INT8 or FP32
- Python examples to illustrate the use of models
- Automated performance comparison



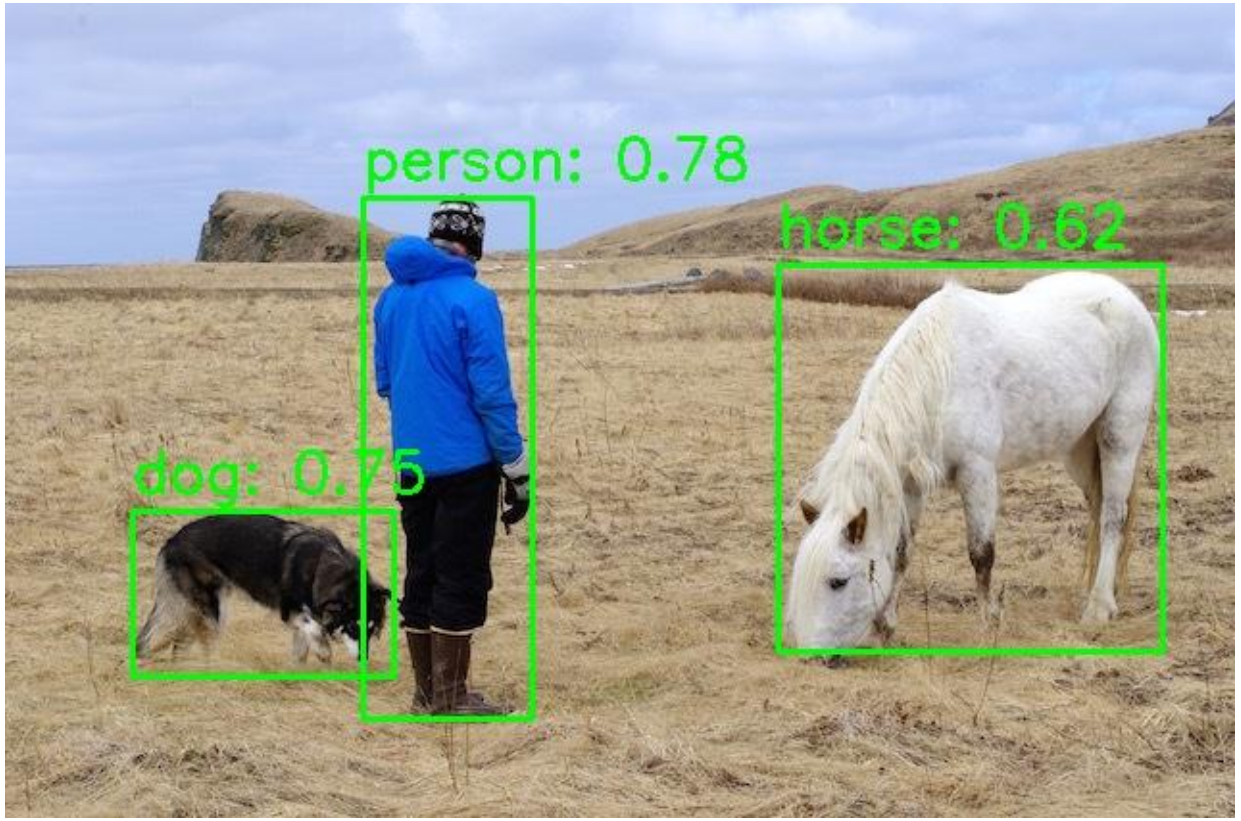
face detection (yunet) and face expression recognition (progressive teacher)



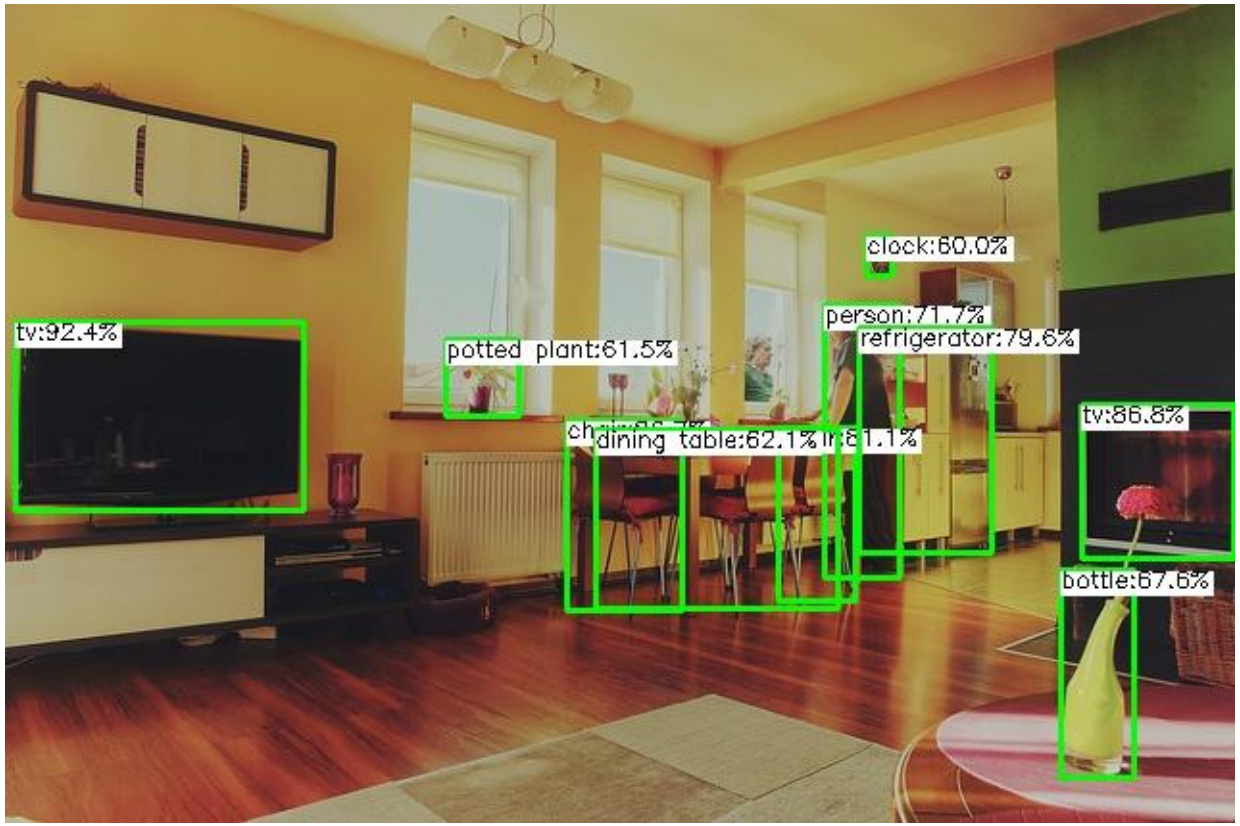
face detection (yunet) and face recognition (sface)

license plate detetion (yunet)

human segmentation (pphumanseg)



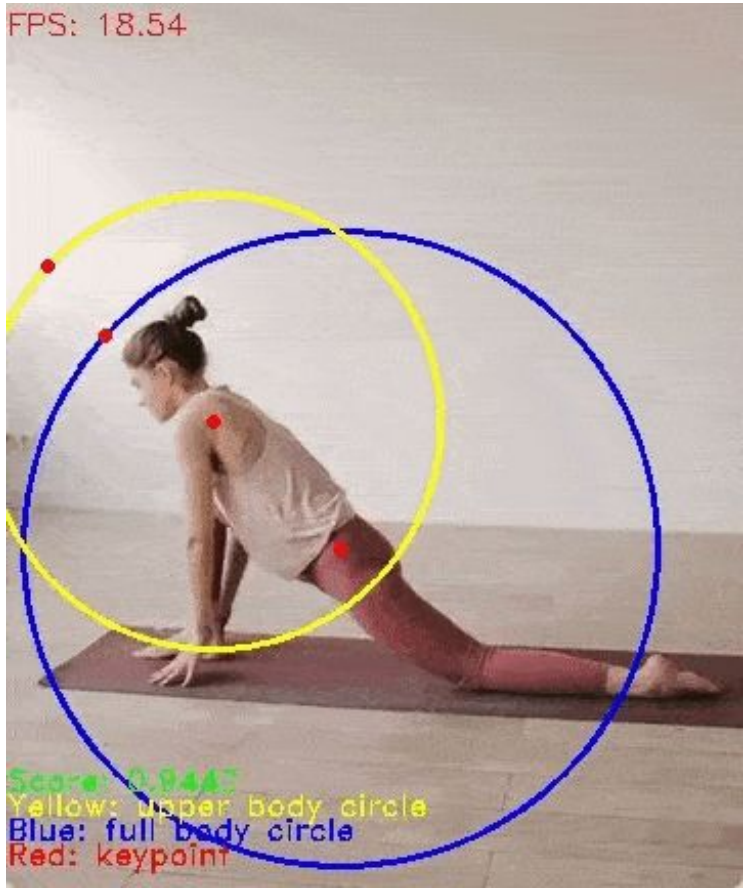
(nanodet)



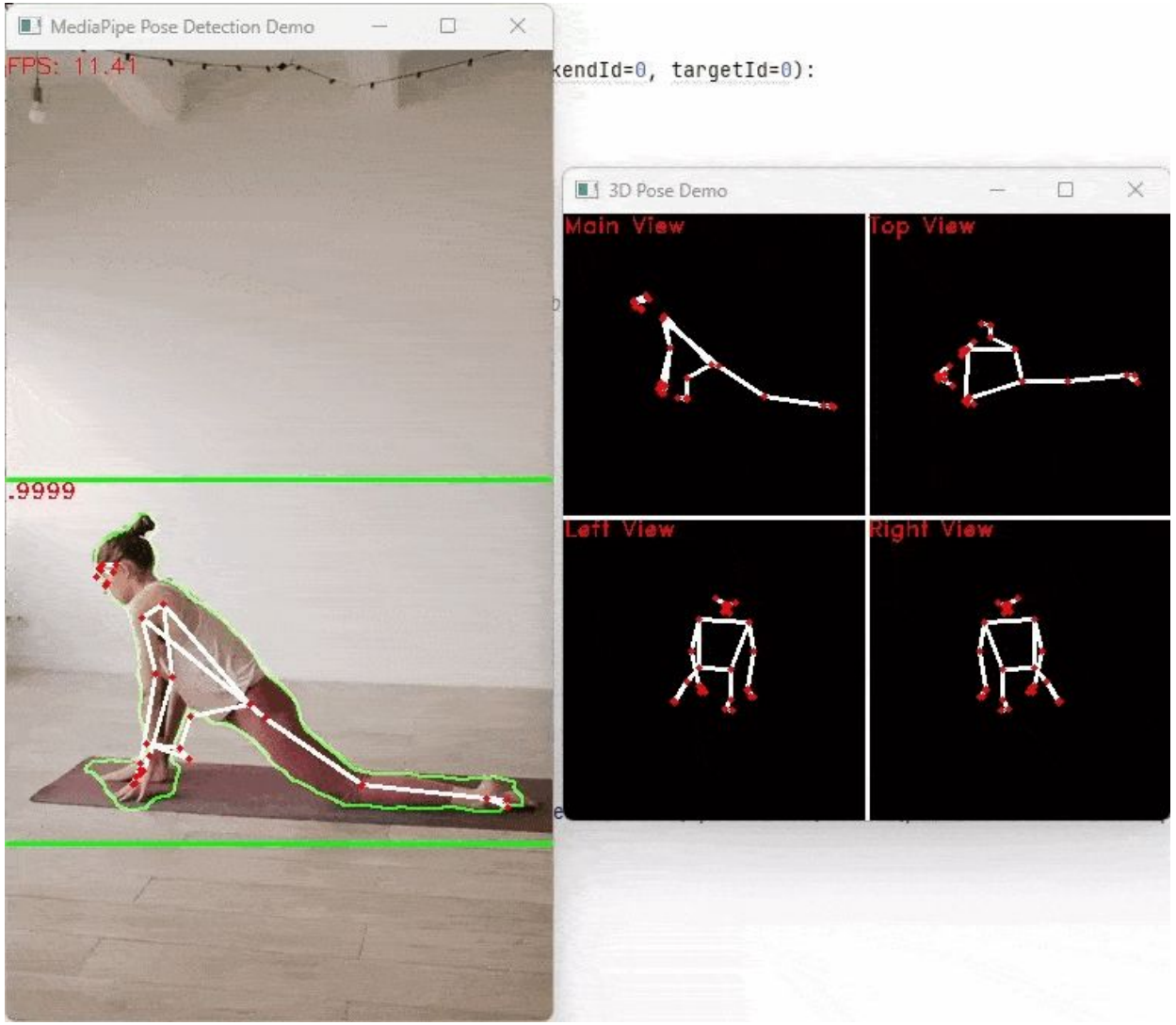
(yolox)

object detection

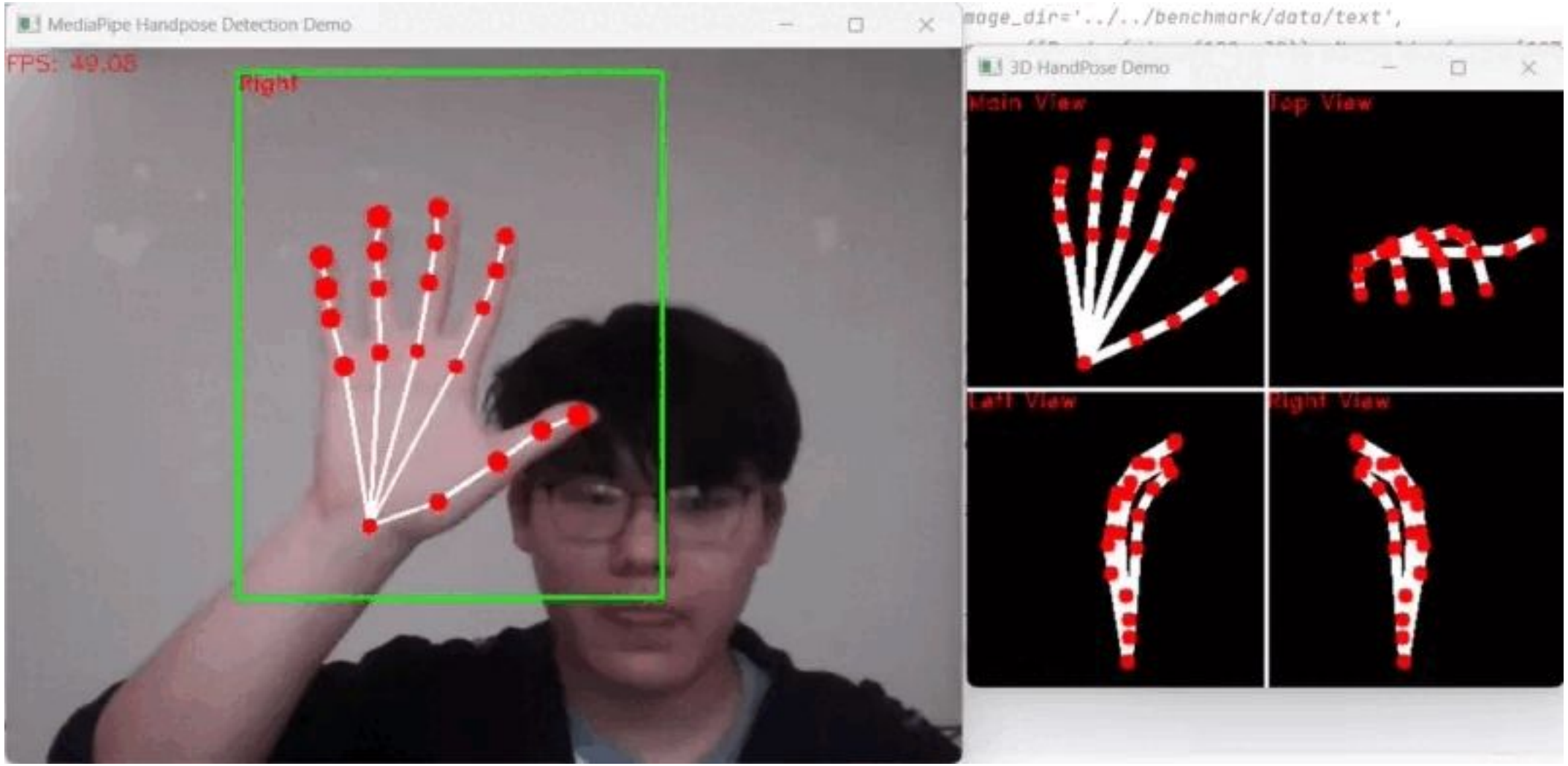
object tracking (vittrack)



persion detection (mp-persondet)



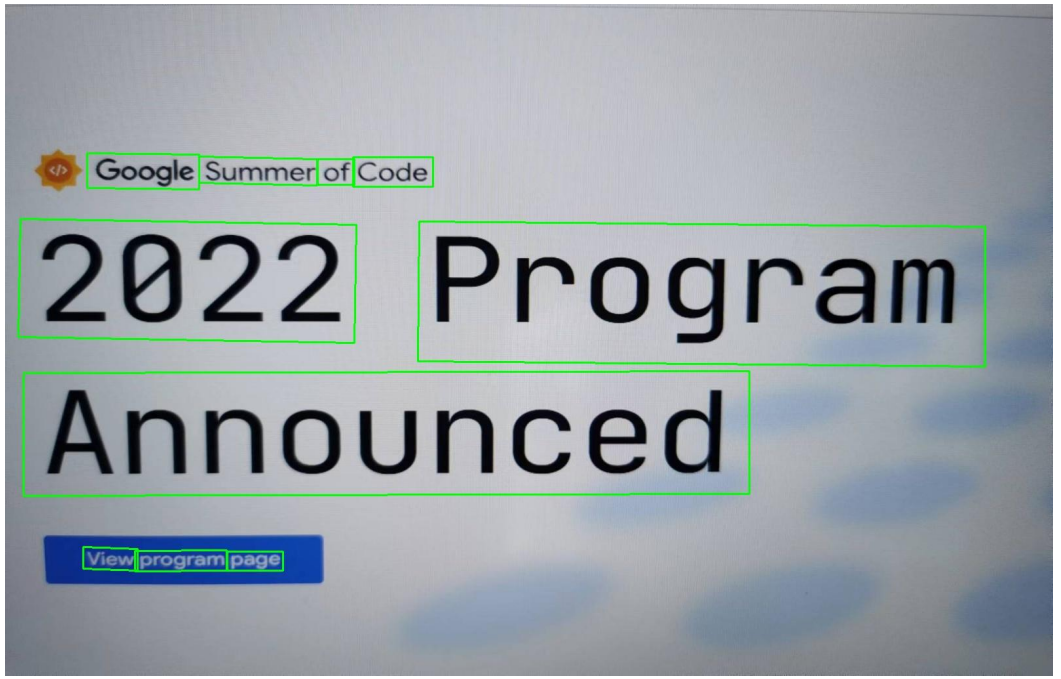
pose estimation (mp-pose)



hand pose estimation (mp-handpose)

QR code recognition (wechatqrcode)

Chinese



English

(ppocr-det)

Models & Benchmark Results

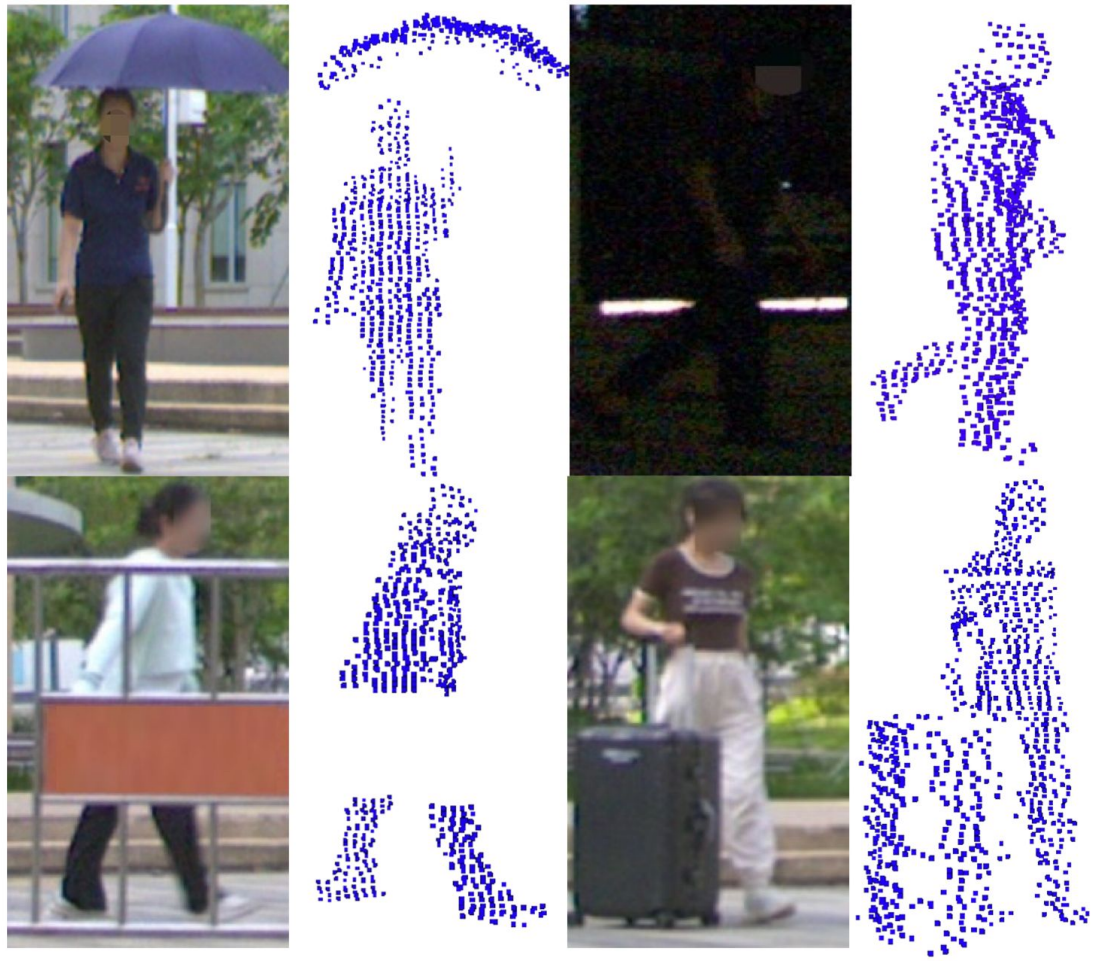
Faster ← → Slower

Model	Task	Input Size	Intel 12700K CPU	Atlas 200I DK A2 Ascend 310B CPU	Atlas 200 DK Ascend 310 CPU	Khadas VIM3 A311D CPU	Khadas VIM4 A311D2 CPU	Khadas Edge2 RK3588S CPU	Jetson Nano B01 CPU	Jetson Nano Orin CPU	Raspberry Pi 4B BCM2711 CPU	Horizon Sunrise Pi X3 CPU	MAIX-III AX-Pi AX620A CPU	Toybrick RV1126 CPU	StarFive VisionFive 2 StarFive JH7110 CPU	Jetson Nano B01 GPU	Jetson Nano Orin GPU	Khadas VIM3 A311D NPU	Atlas 200 DK Ascend 310 NPU
YuNet	Face Detection	160x120	0.69	6.67	7.82	4.62	4.27	2.30	5.62	2.59	6.23	10.56	83.95	56.78	41.13	10.99	5.23	5.24	2.24
SFace	Face Recognition	112x112	5.09	78.90	92.21	55.04	39.94	28.94	64.80	20.05	68.82	124.80	2326.96	1737.74	1169.96	25.25	7.59	45.96	2.66
FER	Face Expression Recognition	112x112	1.79	36.94	42.93	29.50	17.28	12.44	26.54	9.15	27.81	55.12	823.42	609.51	423.91	13.97	8.48	30.25	2.19
LPD_YuNet	License Plate Detection	320x240	5.68	155.73	201.99	128.58	81.46	53.83	134.36	39.91	153.87	252.05	4371.75	3260.22	2267.96	54.88	16.33	32.11	7.63
YOLOX	Object Detection	640x640	78.77	1407.75	1875.33	1108.12	805.54	554.30	1209.12	400.91	1614.13	2516.91	50633.38	37600.81	26185.41	371.32	103.28	408.18	28.59
NanoDet	Object Detection	416x416	41.02	253.05	313.66	179.93	136.14	98.03	215.67	125.31	214.59	399.35	3421.19	2335.89	1731.61	63.86	24.46	116.32	20.62
PPOCRDet-CN	Text Detection	640x480	18.76	219.28	360.26	211.02	148.82	103.42	209.80	78.10	325.02	475.98	5899.23	4267.03	2988.22	112.68	27.49	160.70	---
PPOCRDet-EN	Text Detection	640x480	18.59	217.18	361.22	210.19	148.91	103.41	209.60	78.03	323.54	475.90	5889.39	4265.58	2981.84	112.48	27.53	160.47	---
CRNN-EN	Text Recognition	100x32	9.85	158.82	289.82	185.45	127.49	67.15	245.18	134.47	226.09	427.47	3247.56	2217.15	1432.21	36.77	13.58	196.69	---
CRNN-CN	Text Recognition	100x32	11.03	169.22	318.96	197.16	135.27	70.63	259.28	144.53	240.95	453.60	3281.31	2217.08	1425.42	44.97	15.91	197.61	---
PP-ResNet	Image Classification	224x224	19.47	417.31	499.55	335.75	219.81	151.10	346.78	102.30	420.93	653.39	15841.89	11855.64	8116.21	98.80	32.58	73.19	6.99
MobileNet-V1	Image Classification	224x224	3.13	70.20	92.66	59.27	38.73	27.45	65.72	18.47	64.20	119.71	2123.08	1567.09	1079.69	28.96	8.41	148.80*	5.15
MobileNet-V2	Image Classification	224x224	3.04	61.72	79.39	52.17	33.68	22.95	56.66	17.08	57.91	102.57	1619.08	1188.83	820.15	28.61	9.36	143.17*	5.41
PP-HumanSeg	Human Segmentation	192x192	5.59	78.01	102.49	71.92	47.68	29.63	64.95	24.86	71.35	132.95	2175.34	1109.61	1127.61	67.25	12.91	28.75	6.94
WeChatQRCode	QR Code Detection and Parsing	100x100	1.19	---	---	---	---	---	5.56	3.53	5.90	9.93	---	---	---	---	---	---	---
YoutuReID	Person Re-Identification	128x256	15.56	285.75	521.46	327.07	218.22	148.01	356.78	93.58	478.89	678.74	14895.99	11143.23	7613.64	90.84	23.53	41.82	5.58
MP-PalmDet	Palm Detection	192x192	5.35	92.56	159.80	78.26	54.45	45.03	75.30	27.73	97.05	168.54	1285.54	905.77	628.64	37.61	10.84	37.34	5.17
MP-HandPose	Hand Pose Estimation	224x224	2.40	56.00	67.85	35.80	25.83	20.22	38.45	15.00	42.58	77.44	664.73	465.12	319.69	24.47	12.29	19.75	6.27
MP-PersonDet	Person Detection	224x224	7.65	90.13	145.83	83.22	57.22	42.90	87.65	33.05	105.60	172.55	1849.87	1315.48	910.38	37.39	14.50	---	16.45
MP-Pose	Pose Estimation	256x256	6.33	83.16	134.02	75.38	53.06	37.06	75.20	31.51	116.15	162.87	1045.98	736.02	524.52	76.44	26.54	---	---
VitTrack	Object Tracking	1280x720	4.01	37.02	143.62	32.20	29.47	14.02	48.39	19.16	48.55	84.15	548.36	411.41	288.95	47.26	19.75	---	---

Units: All data in milliseconds (ms).
*: Models are quantized in per-channel mode, which run slower than per-tensor quantized models on NPU.

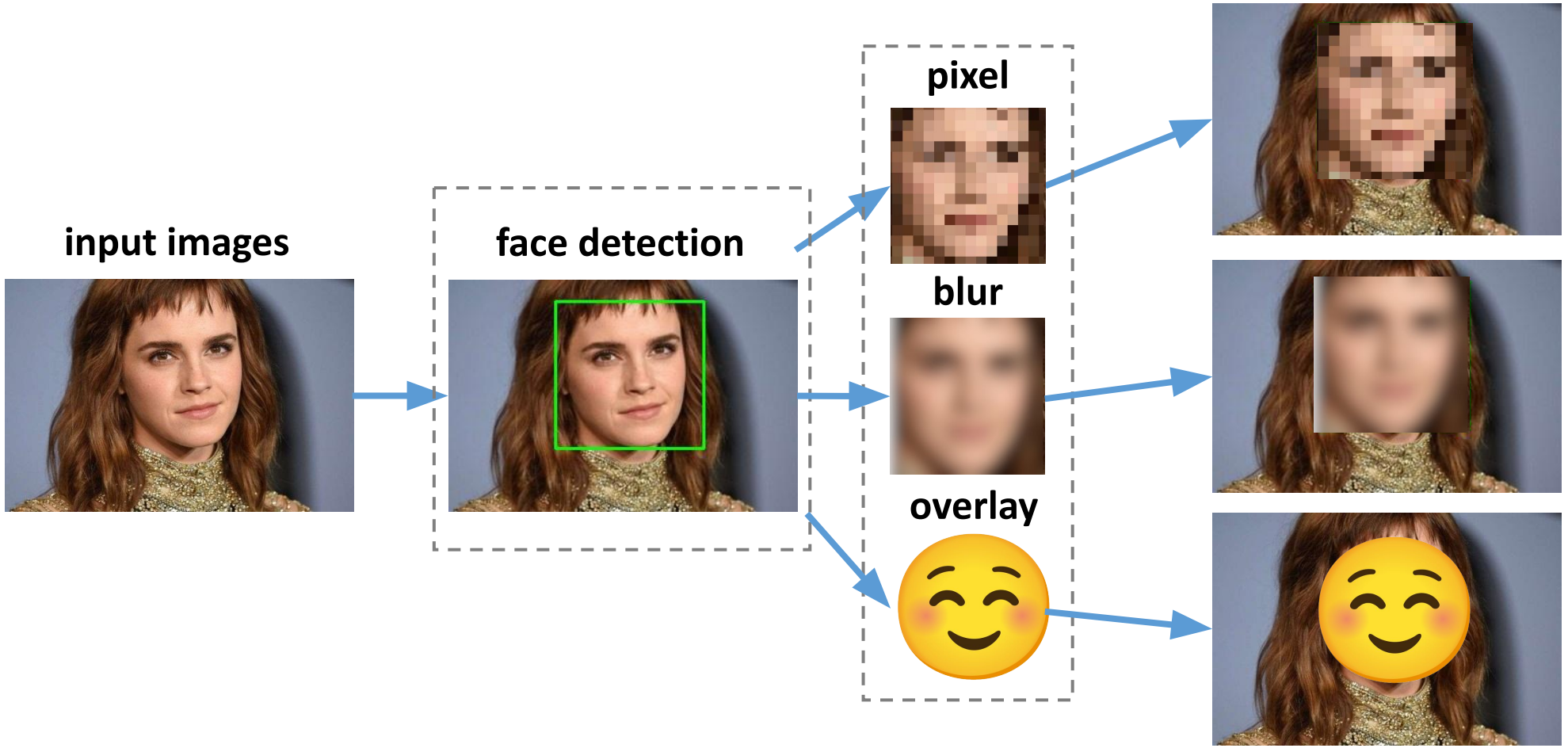
Hardware:
x86-64:
Intel Core i7-12700K
ARM:
Khadas VIM3 Khadas VIM4 Khadas Edge 2 Atlas 200 DK Atlas 200I DK A2 NVIDIA Jetson Nano B01 NVIDIA Jetson Nano Orin Raspberry Pi 4B Horizon Sunrise X3 MAIX-III AXera-Pi Toybrick RV1126
RISC-V:
StarFive VisionFive 2 Allwinner Nezha D1 Xuantie C906 CPU

1. Face Anonymization for Privacy



Diverse attributes in SUSTech1K

Face Anonymization Pipeline



Face Detection

- Face detection with **cv.FaceDetectorYN** :

- Works with YuNet:

  https://github.com/opencv/opencv_zoo -> models -> face_detection_yunet

  - Note: **faces of pixels 10x10 to 300x300 can be detected**.
- Call setInputSize() if the actual input size is different from the initialized.
- Returns a length-2 tuple:
  - results[0]: no use
  - results[1]:
    - None, if no face detected
    - Numpy array of shape: nx15, [[x1, y1, w, h, lm1_x, lm1_y, …, score], …

```python
import numpy as np
import cv2 as cv

# Step 1: load model
net = cv.FaceDetectorYN.create(
  model="face_detection_yunet_2022mar.onnx",
  config="",
  input_size=[480, 640], # [width, height]
  score_threshold=0.99,
  backend_id=cv.dnn.DNN_BACKEND_DEFAULT, # optional
  target_id=cv.dnn.DNN_TARGET_CPU, # optional
)

# Step 2: load and set input
img = cv.imread("/path/to/image")
# (optional): setInputSize if not [480, 640]
h, w, c = img.shape
net.setInputSize([w, h])

# Step 3: detect faces
results = net.detect(img)
faces = results[1]
```

Face Anonymization

- Face Pixelization:
  - Downscale the face region by reducing its resolution
  - Upscale the face region by increasing its resolution
  - by using **resize()** function
- Face Blur:
  - blur the face by **GaussianBlur()**
- Face Overlay:
  - resize the mask image and put it on face

29

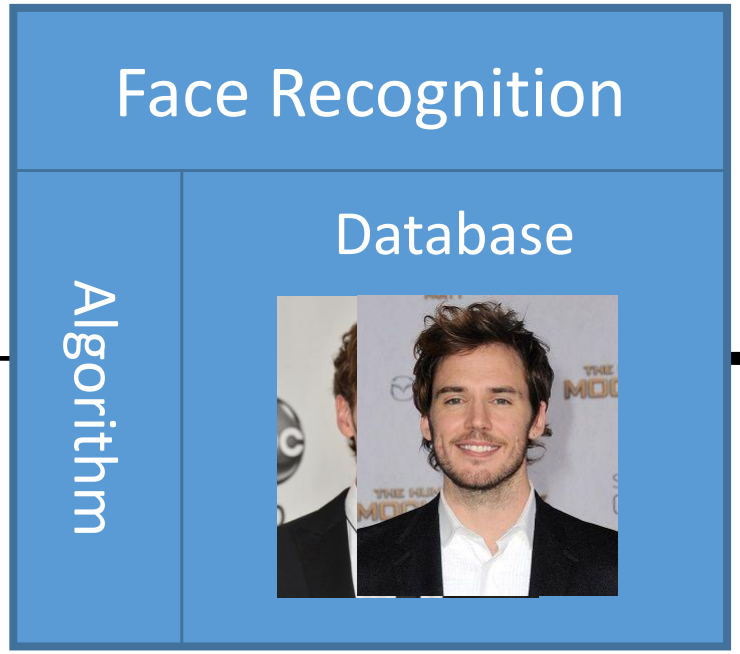
2. Build a face recognition system

- Face recognition in brief
- How to build the FR system using OpenCV
- Assignment

What is face recognition?

A B C

Input



A: Found in database

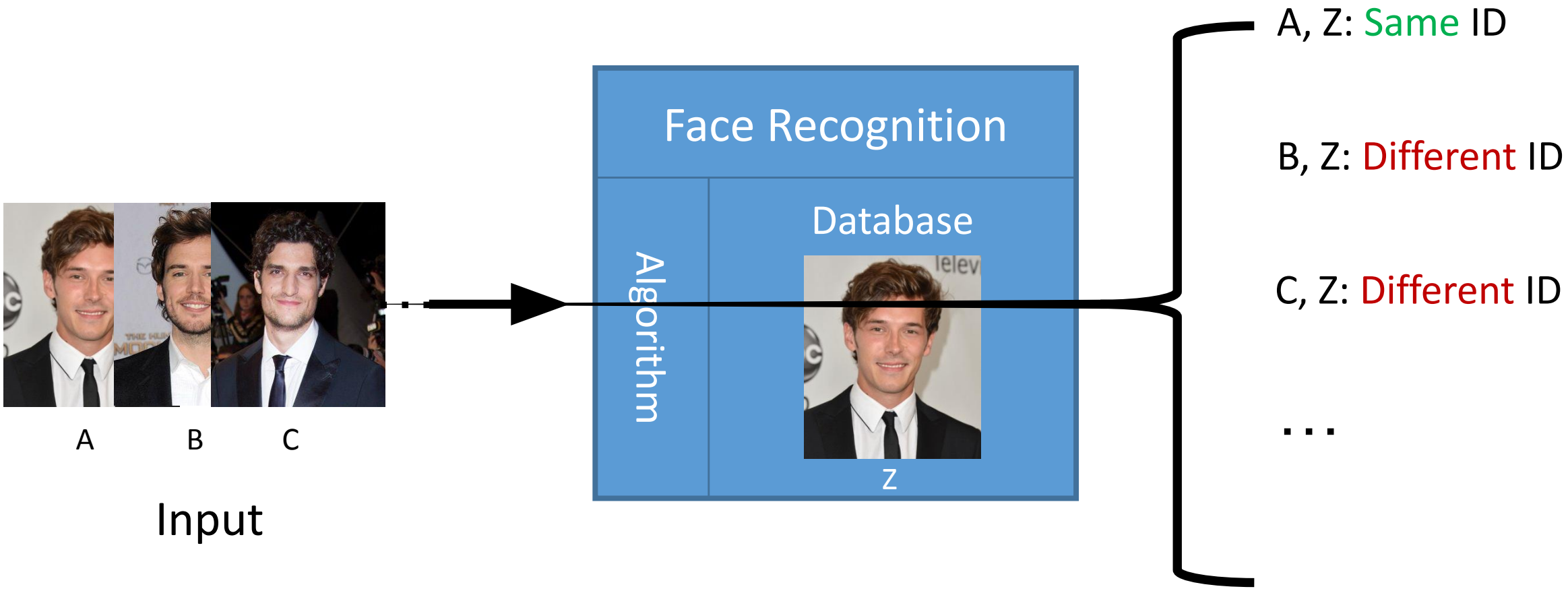
B: Found in database

C: Not Found

...

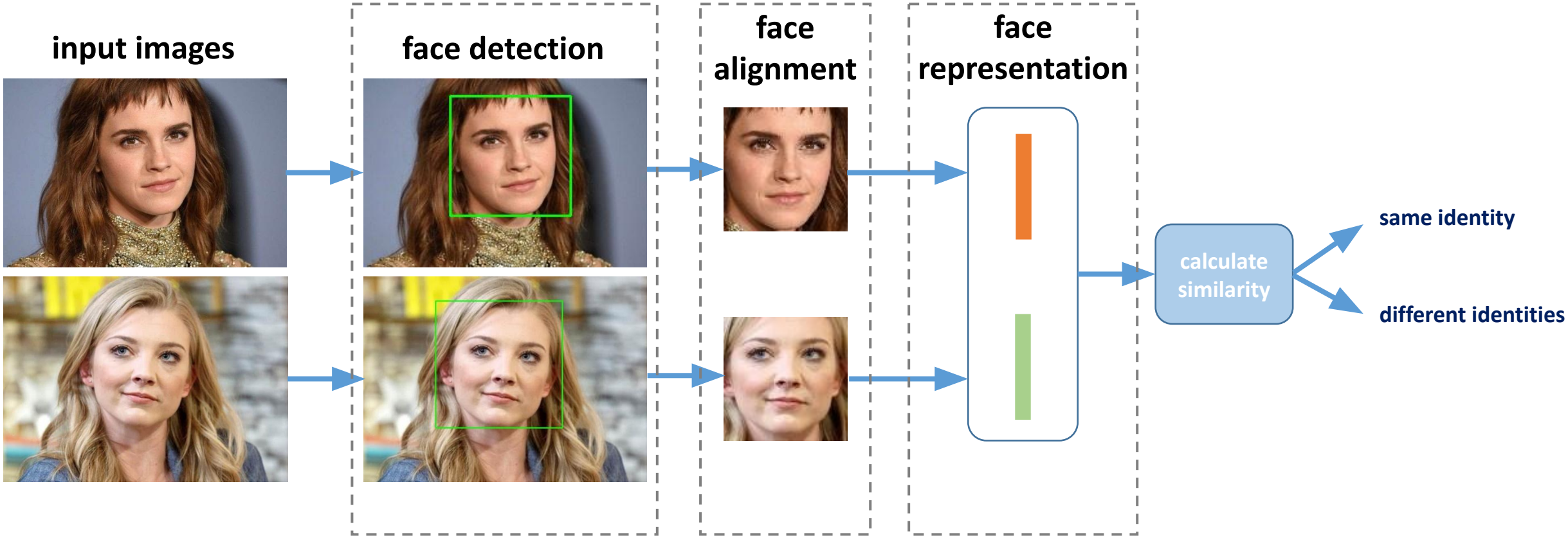
1:N Face recognition: Is the input found in database?

What is face recognition?

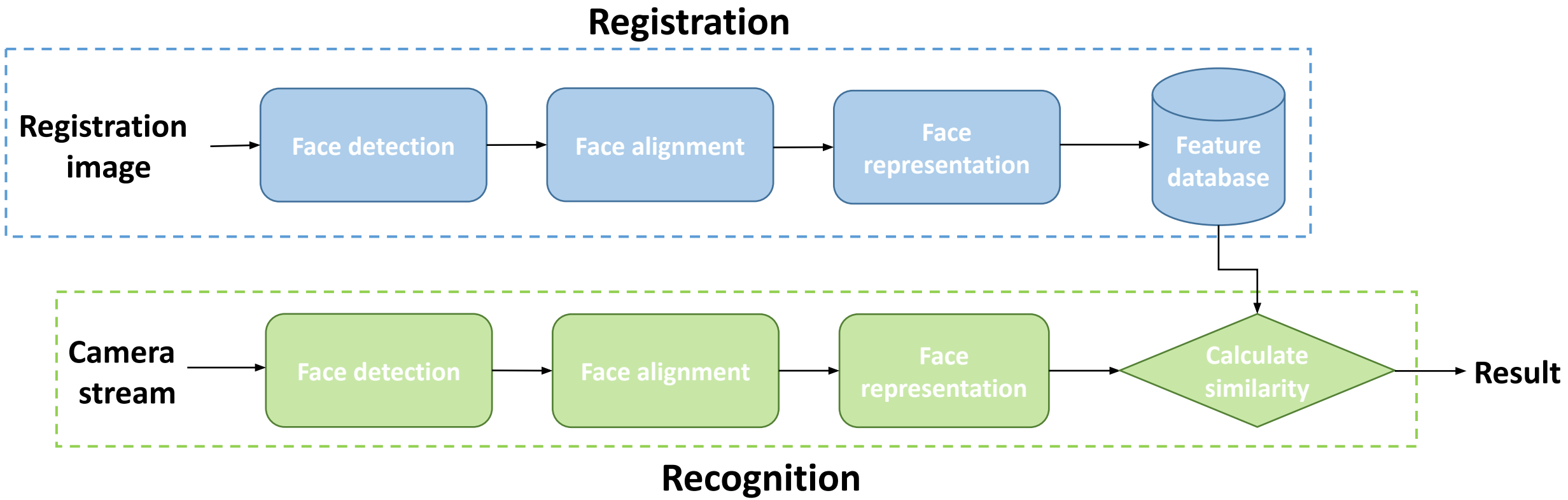


1:1 Face recognition: Does the input have same ID as Z?

Face Recognition System Pipeline



Face recognition system: architecture

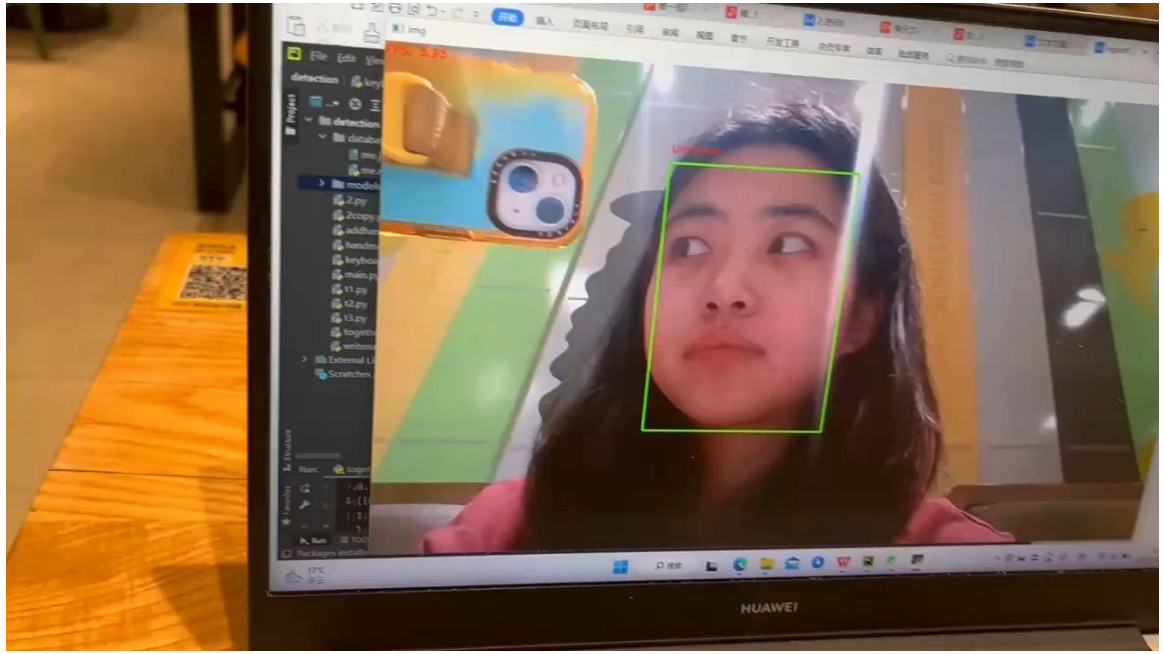


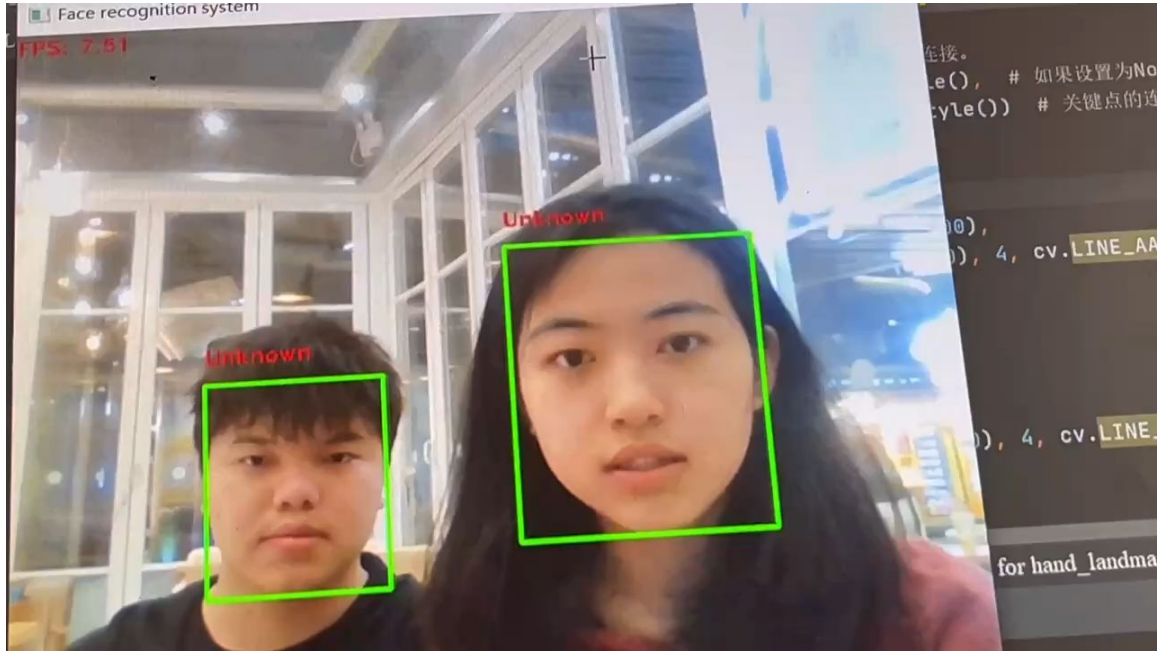
Building the FR system

- Face recognition with **cv.dnn.FaceRecognizerSF**
  - Works with SFace: https://github.com/opencv/opencv_zoo -> models -> face_recognition_sface
  - Call **cv.FaceRecognizerSF.alignCrop** with the image and the face bounding box to get aligned face crop.
  - Call **cv.FaceRecognizerSF.feature** with the aligned face to extract face feature.
  - Call **cv.FaceRecognizerSF.match** with two extracted features to calculate score for match.
    - match(f1, f2, 0): cosine similarity, threshold **0.363**
    - match(f1, f2, 1): L2 distance, threshould **1.128**

```python
import numpy as np
import cv2 as cv

# Step 1: load model
yunet = cv.FaceDetectorYN.create(
  model="face_detection_yunet_2022mar.onnx",
  config="",
  input_size=[480, 640], # [width, height]
  score_threshold=0.99,
  backend_id=cv.dnn.DNN_BACKEND_DEFAULT, # optional
  target_id=cv.dnn.DNN_TARGET_CPU, # optional
)
sface = cv.FaceRecognizerSF.create(
  model="face_recognition_sface_2021dec.onnx",
  config="",
  backend_id=cv.dnn.DNN_BACKEND_DEFAULT, # optional
  target_id=cv.dnn.DNN_TARGET_CPU, # optional
)

# Step 2: load image
img1 = cv.imread("/path/to/image1")
img2 = cv.imread("/path/to/image2")

# Step 3: detect faces, align, extract features and match
faces1 = net.detect(img1)[1]
face1 = faces1[0][:-1] # take the first face and filter out score
faces2 = net.detect(img2)[1]
face2 = faces2[0][:-1]
aligned_face1 = sface.alignCrop(img, face1)
aligned_face2 = sface.alignCrop(img, face2)
feature1 = sface.feature(aligned_face1)
feature2 = sface.feature(aligned_face2)
score = sface.match(feature1, feature2)
```

face+gesture recognition





WSB2023 Yangruohong,Zhanghui

Assignment

A. Face Anonymization System (**don't need to submit anything, just for practice**)

B. Build a **1:N face recognition system** with **camera stream as input** **using OpenCV**. Write a report about your system in English .

C. Build **any other kind of biometric recognition system** using OpenCV. Write a report about your system in English .

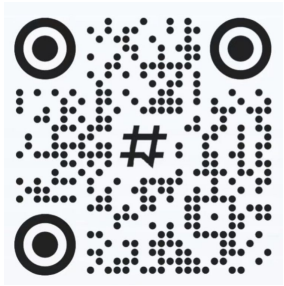
Notes:

- Choose B or C for assignment. Team up with students from **different universities** .
- Reference: https://github.com/OpenCVChina/WSB-assignment.
- Bonus:
  - Combine face anonymization with face recognition.
  - Hand pose recognition for registering unknown identities / removing identities from database.
  - Other creative interactions using deep learning models …
- **Send your report & code to: wanli.zhong@opencv.org.cn** (subject line: [WSB] TeamName-ProjectTitle)
- **Deadline: 2025-01-15 23:59:59 (UTC+8)**
- Awards will be given to the top-3 teams.

Thank You!




QQ频道



OpenCV China

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY