Capstone Project
Edge computing device programming for AI projects

Lecture 4

Dr. Wilton Fok &

Carol Chen

Recap: Training classification model

	Operation	Results
Dataset preparation	**(Preparing the catdog classification dataset)**	**Get a prepared dataset**
Train model	***cd ~/jetson-inference/python/training/classification python3 train.py --model-dir=models/cat_dog data/cat_dog --batch-size=1 --workers=1 --epochs=1***	**Obtaining a .pth model**
Convert model	***python3 onnx_export.py --input=model_best.pth.tar --model-dir=models/cat_dog***	**Obtaining a .onnx model**
Run model	***/home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model= models/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=data/cat_dog/labels.txt data/cat_dog/test/cat***	**Visualize the output of the model**

GRAMMAR and TRANSLATION of terminal command

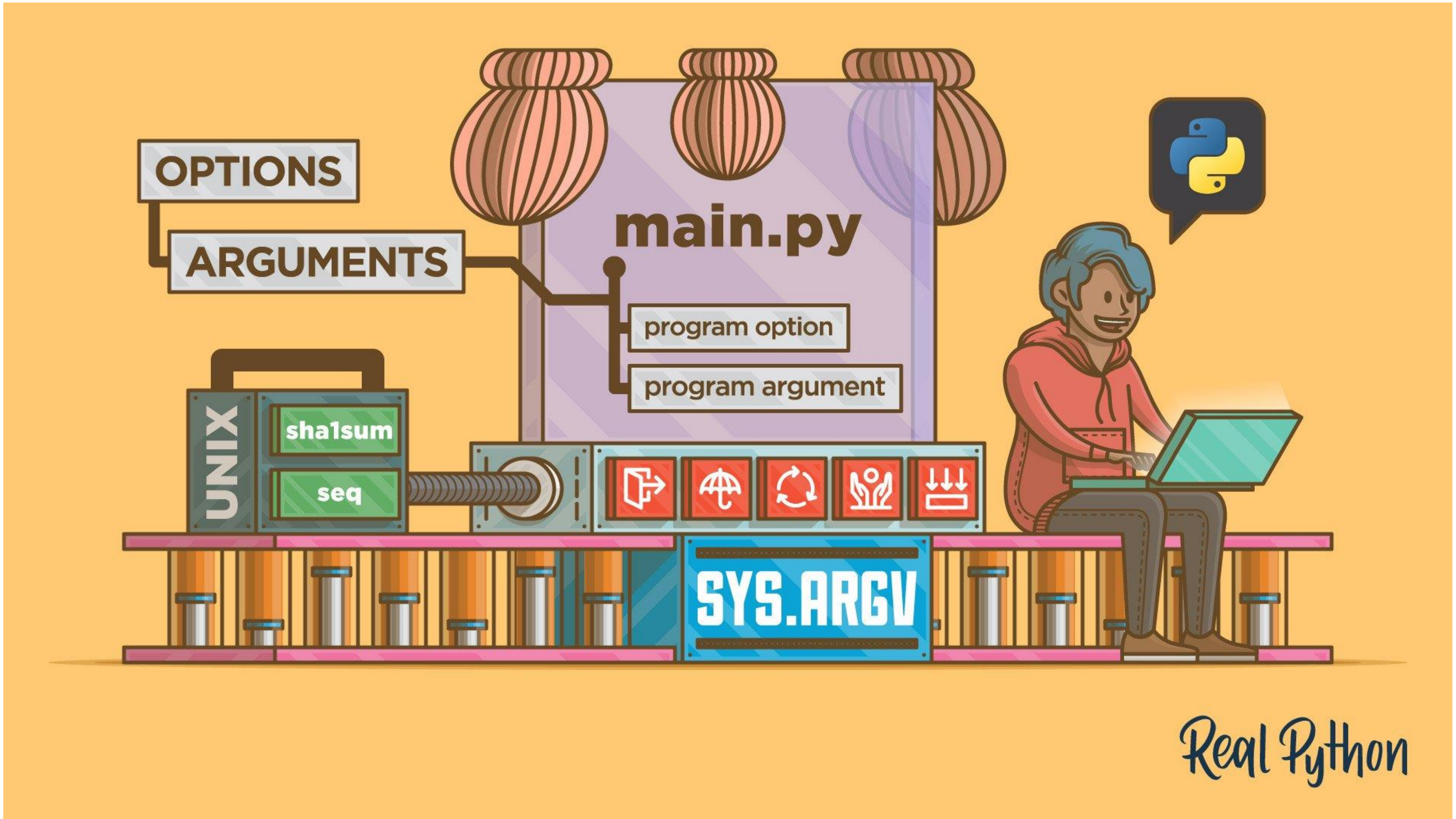
Blue: predicate (verb) **Green: object (noun)** **Purple: adverbial // arguments**

	Operation	Translation
Entering target folder	*cd ~/jetson-inference/python/training/classification*	*Told the system to Change the directory into ~/jetson-inference/python/training/classification*
Train model	*python3 train.py --model-dir=models/cat_dog data/cat_dog --batch-size=1 --workers=1 --epochs=1*	*Use the software python3 to run the file "train.py" with the following arguments "--model-dir=models/cat_dog data/cat_dog --batch-size=1 --workers=1 --epochs=1"*
Convert model	*python3 onnx_export.py --input=model_best.pth.tar --model-dir=models/cat_dog*	*Use the software python3 to run the file onnx_export.py* using the model models/cat_dog
Run model	*/home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model= models/resnet18.onnx --input_blob=input_0 --output_blob=output_0 –labels=data/cat_dog/labels.txt data/cat_dog/test/cat save/cat*	Directly Execute the file "/home/nvidia/jetson-inference/build/aarch64/bin/imagenet" with the following arguments "*--model=models/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels data/cat_dog/labels.txt data/cat_dog/test/cat save/cat*"

For each elements, we should add a space to separate them; While finish a sentence, using Enter for ending.

What’s the arguments?

Argument means the parameters you use while executing something.



What’s the arguments?

Command in terminal
python3 train.py --model-dir=models/cat_dog data/cat_dog --batch-size=1 --workers=1 --epochs=1
/home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model= models/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=data/cat_dog/labels.txt data/cat_dog/test/cat save/cat

This library allows us to pass parameters into the program directly from the command line. We can use python file.py to run python files. The function of argparse is to parse, save and use other parameters passed in from the command line.

.py file

```
parser.add_argument("input_URI", type=str, default="", nargs='?', help="URI of the input stream")
parser.add_argument("output_URI", type=str, default="", nargs='?', help="URI of the output stream")
parser.add_argument("--network", type=str, default="resnet18-body", help="pre-trained model to load (see below for options)
parser.add_argument("--overlay", type=str, default="links,keypoints", help="pose overlay flags (e.g. --overlay=links,keypoi
parser.add_argument("--threshold", type=float, default=0.15, help="minimum detection threshold to use")

# parser.add_argument("--keypoint1", type=str, default="", help="when calculating the angle, choose the first keypoint")
# parser.add_argument("--keypoint2", type=str, default="", help="when calculating the angle, choose the second keypoint")
# parser.add_argument("--keypoint3", type=str, default="", help="when calculating the angle, choose the third keypoint")
```

What's the arguments?

- Argument means the parameters you use while executing something.

Command	Translation
python3 train.py --model-dir=models/cat_dog data/cat_dog --batch-size=1 --workers=1 --epochs=1	***we are telling the system that for this training, we need to "**store the model in models/cat_dog, using the data from data/cat_dog. The training uses 1 batchsize and 1 workers. We train the model for 1 epochs.**"***
/home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model=models/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=data/cat_dog/labels.txt data/cat_dog/test/cat save/cat	***we are*** **telling the system that while running this fille, "I am using the model from *models/resnet18.onnx*, and its input_blob and output_blob are *input_0* and *output_0* respectively. Also, I want to processing the images in the *data/cat_dog/test/cat* folder, and the corresponding label is *data/cat_dog/labels.txt data/cat_dog/test/cat*. Please save the processed images into *save/cat*"**

Grammar: For each argument, we use double dash **-- to specify, = to assign, and space for termination**.
For example, we want to set model-dir to be "models/self", batch-size to be 1, then the command will be:

--model-dir=models/self --batch-size=1

Remember a space here

How to achieve AI training?

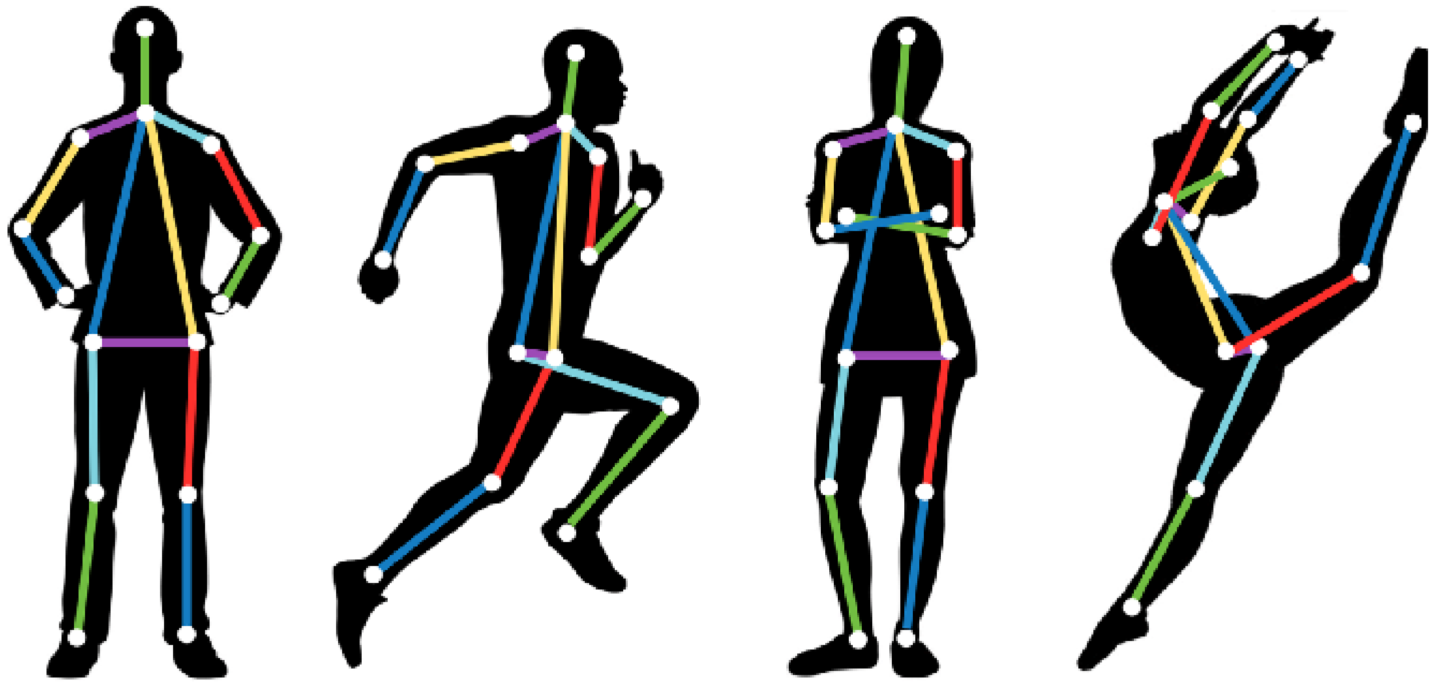
training

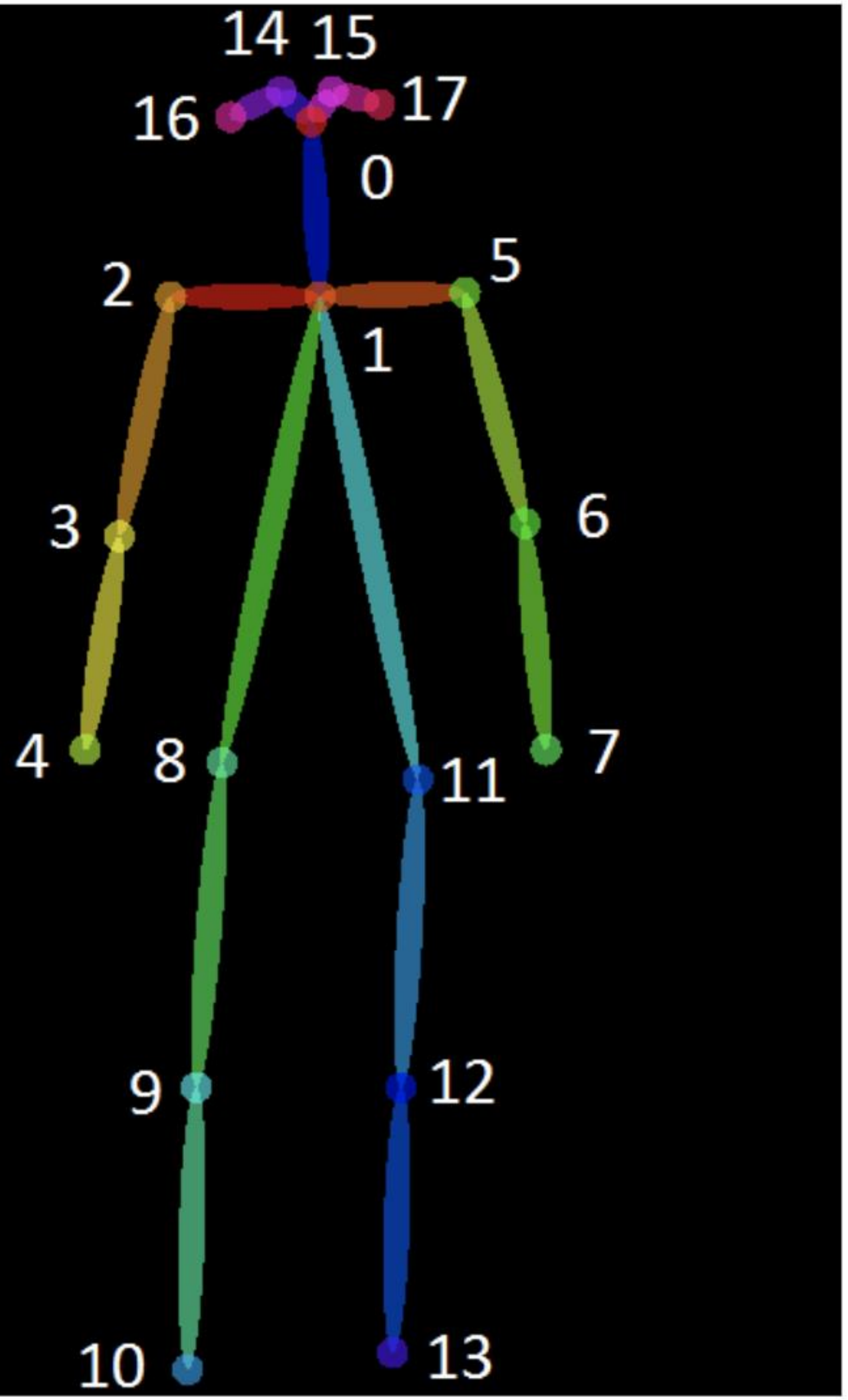
python3 train.py --model-dir=models/cat_dog data/cat_dog --batch-size=1 --workers=1 --epochs=1

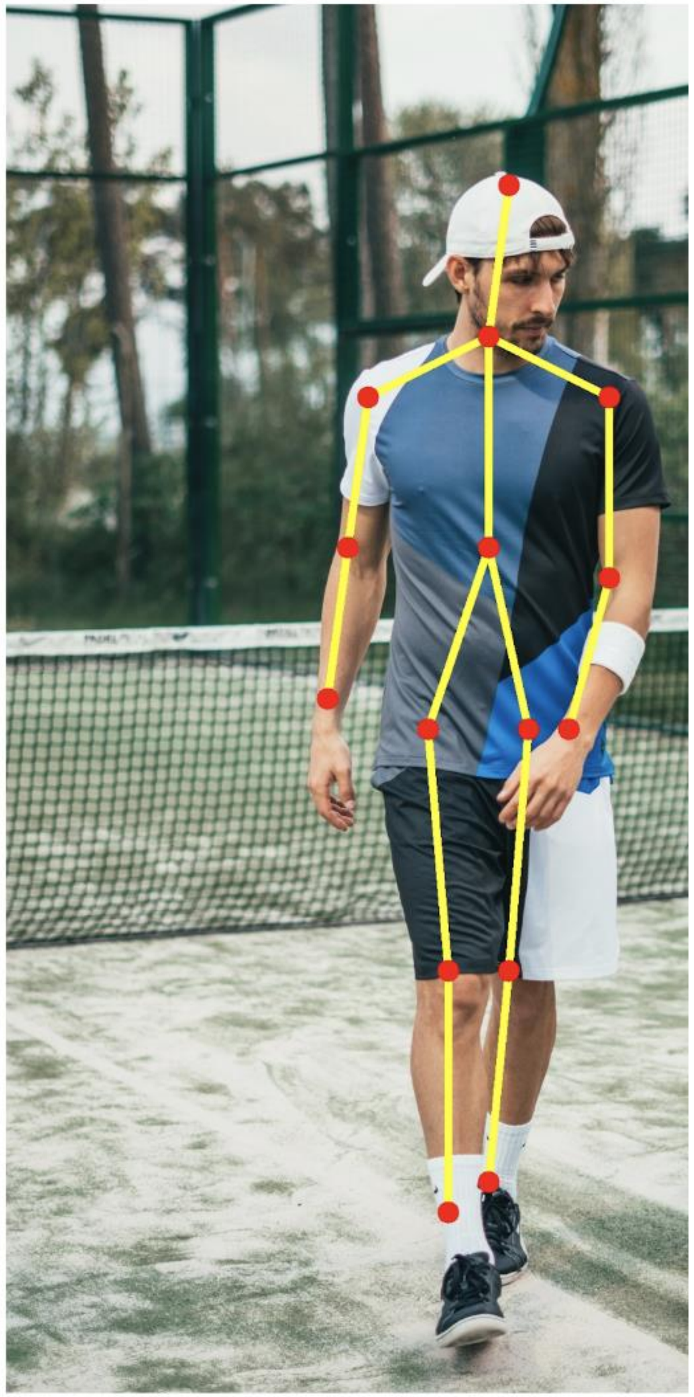
inference

/home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model= models/resnet18.onnx --input_blob=input_0 --output_blob=output_0 –labels=data/cat_dog/labels.txt data/cat_dog/test/cat save/cat

Pose estimation





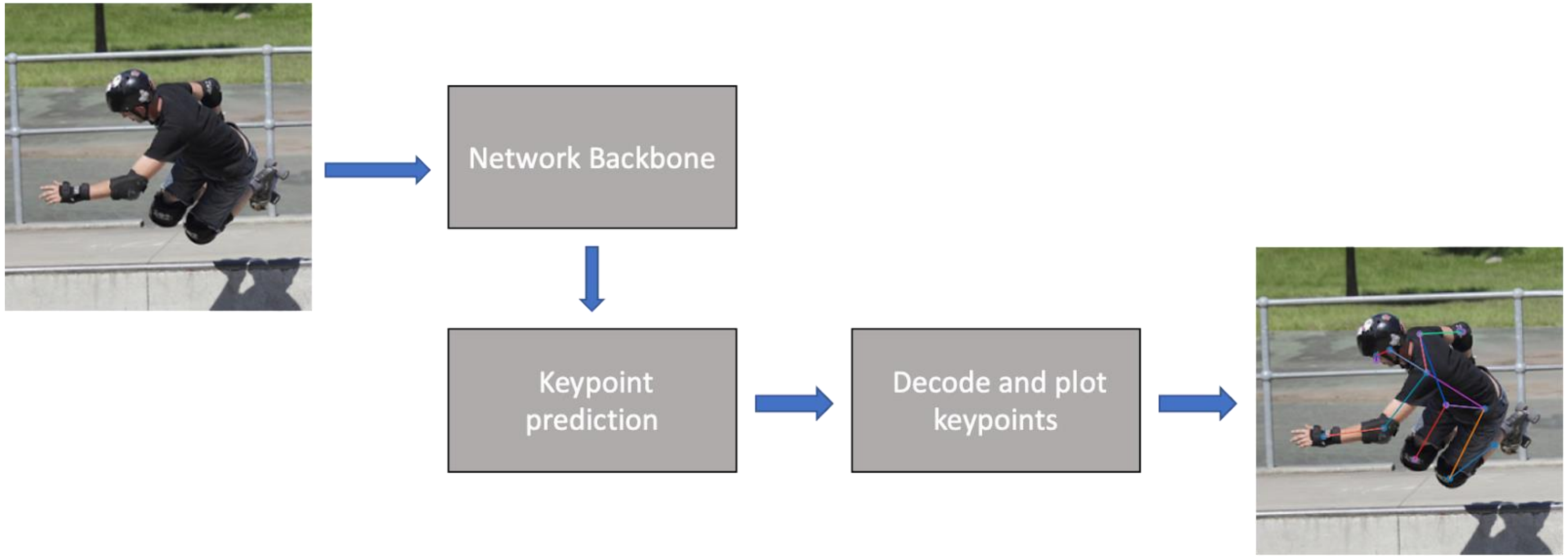


Pose estimation-Introduction

Human Pose Estimation identifies the poses of human body parts and joints in images or videos.

Pose estimation- Introduction

When an image or video is given to the pose estimator model as input, it identifies the coordinates of the detected body parts like wrist, shoulder, knees, eyes, arms and so on, then joints as output and a confidence score showing precision of the estimations.

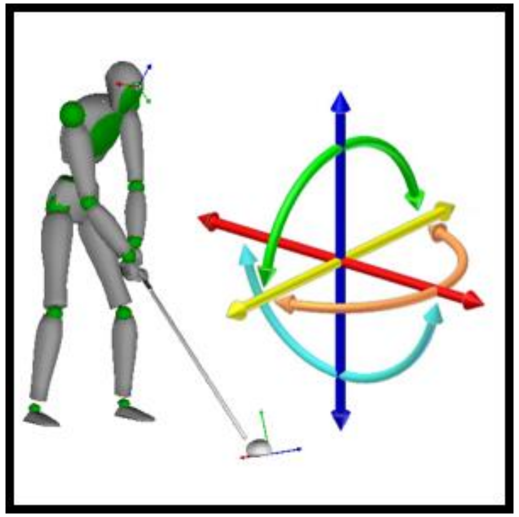


The architecture of Human Pose Estimation

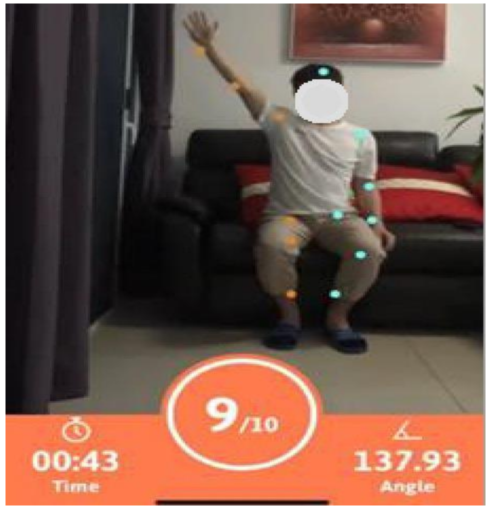
Pose estimation-Application

Human pose estimation is widely used in many applications, here are a few common ones:

Human-computer interaction, by analyzing human posture and movements, enables non-contact human-computer interaction.



Motion analysis can analysis the human body's pose and movements to analyze the body's movement state and trajectory, and do the motion correction



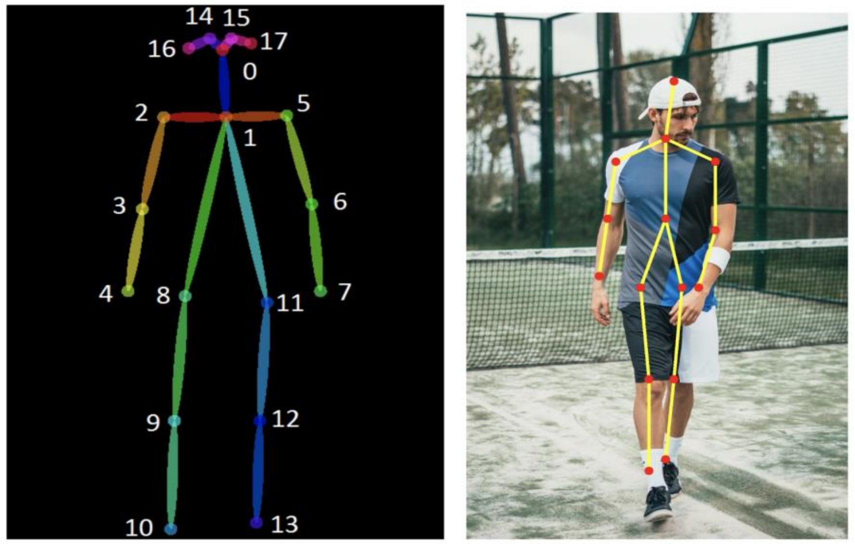
Medical Rehabilitation helps Patients with Stroke Recovery by estimating their movements

Different kinds of pose estimation

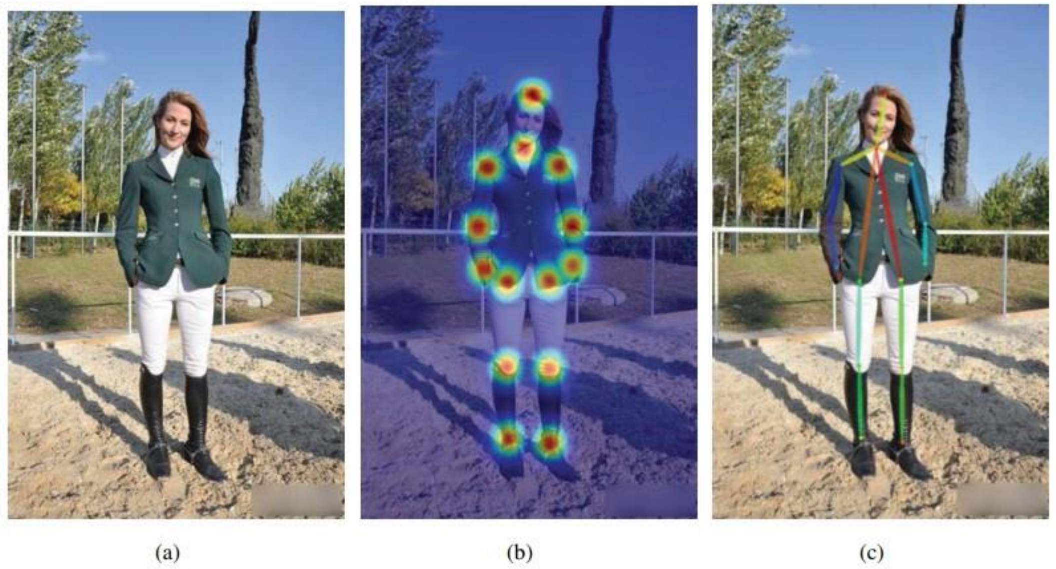
- 2D pose estimation

2D pose estimation uses a single camera or image to predict the location of keypoints.

There are two representations of keypoint locations: **coordinate** and **heatmap**. The coordinate representation only needs to regress the location of each keypoint, while the heatmap representation requires the network to predict the probability that each pixel point belongs to a certain type of keypoint.



Coordinate representation

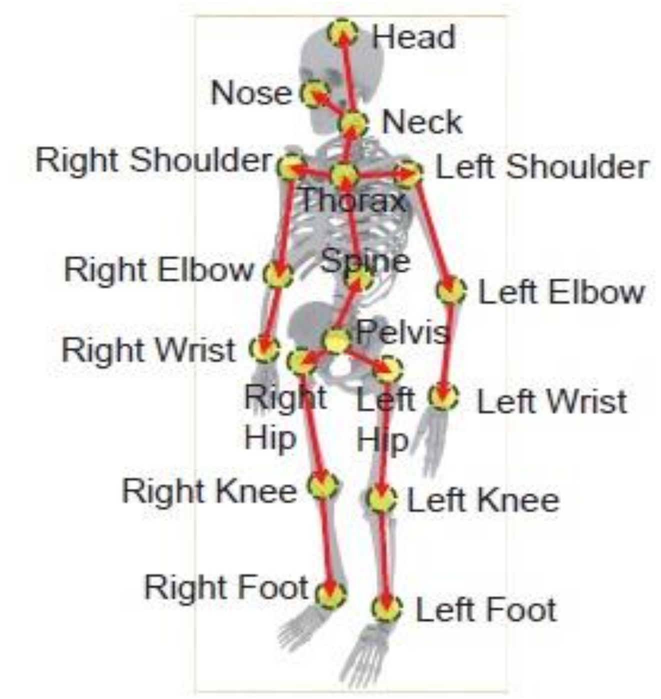


Heatmap representation

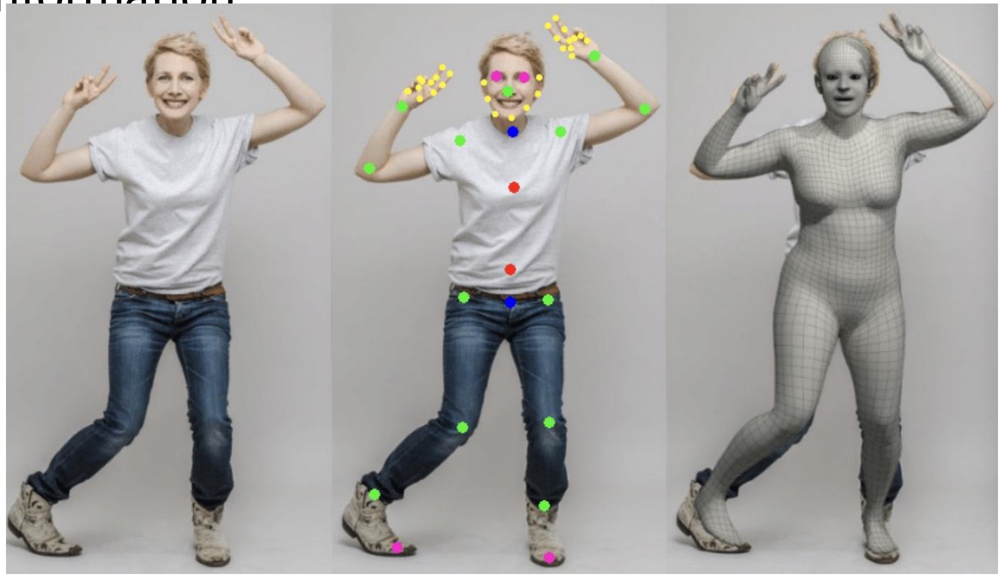
- 3D pose estimation

3D pose estimation uses multiple cameras or image sequences to infer the 3D pose of the human body. Such methods usually require camera calibration and correlation of multiple viewpoints to recover the 3D pose of the human body. There are two main forms of output of 3D HPE prediction:

1) 3D skeleton, which is similar to 2D, except that the skeleton information is presented in a three-dimensional coordinate system



2) 3D shape or mesh, which uses either a parametric human body model or a triangulated mesh to present the predicted results, and which is a more dense skeleton information



Some popular approaches for pose estimation

- **SMPL: A Skinned Multi-Person Linear Model （2015）**

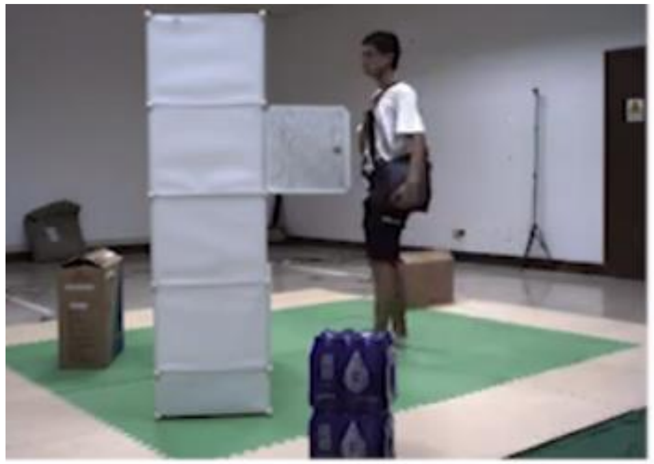
A standard 3D human model, the main parameters are shape blend shapes parameters to control the body shape and pose blend shapes parameters to control the body posture.(video link: https://www.youtube.com/watch?v=kuBlUyHeV5U&t=287s&ab_channel=MichaelBlack)

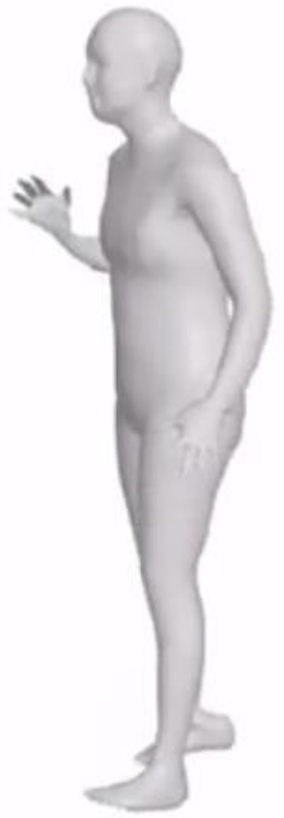
SMPL: A Skinned Multi-Person Linear Model (SIGGRAPH Asia 2015)

Some popular approaches for pose estimation

- **Object-Occlued Human Shape and Pose Estimation from a Single Color Image （2020）**

This research is using the SMPL model as an output representation to extract the 3d mesh of the SMPL model from a single map and also solves the problem of occlusion (video link: https://www.youtube.com/watch?v=8udm2OB0A-U&t=189s&ab_channel=YangangWang)

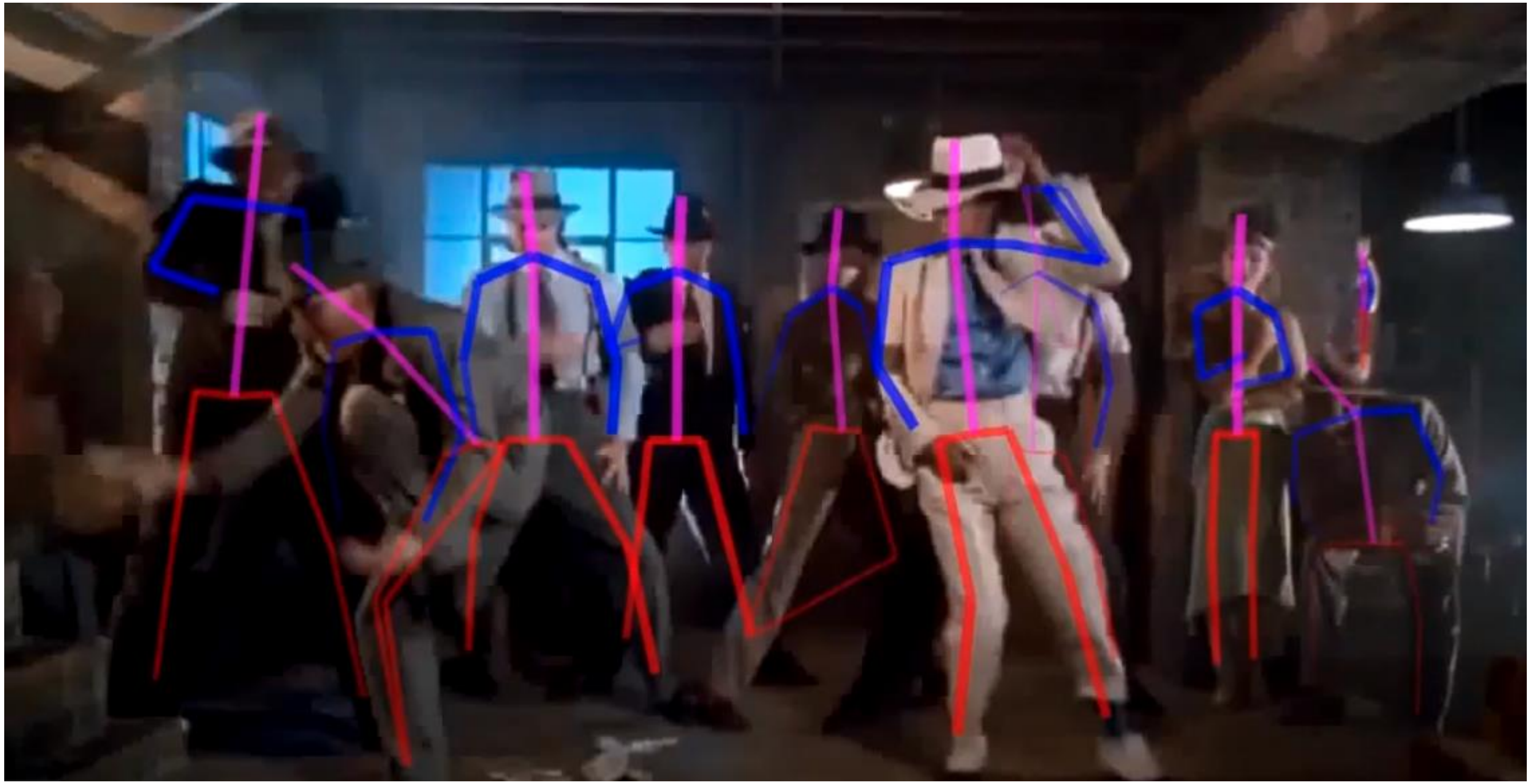




Some popular approaches for pose estimation

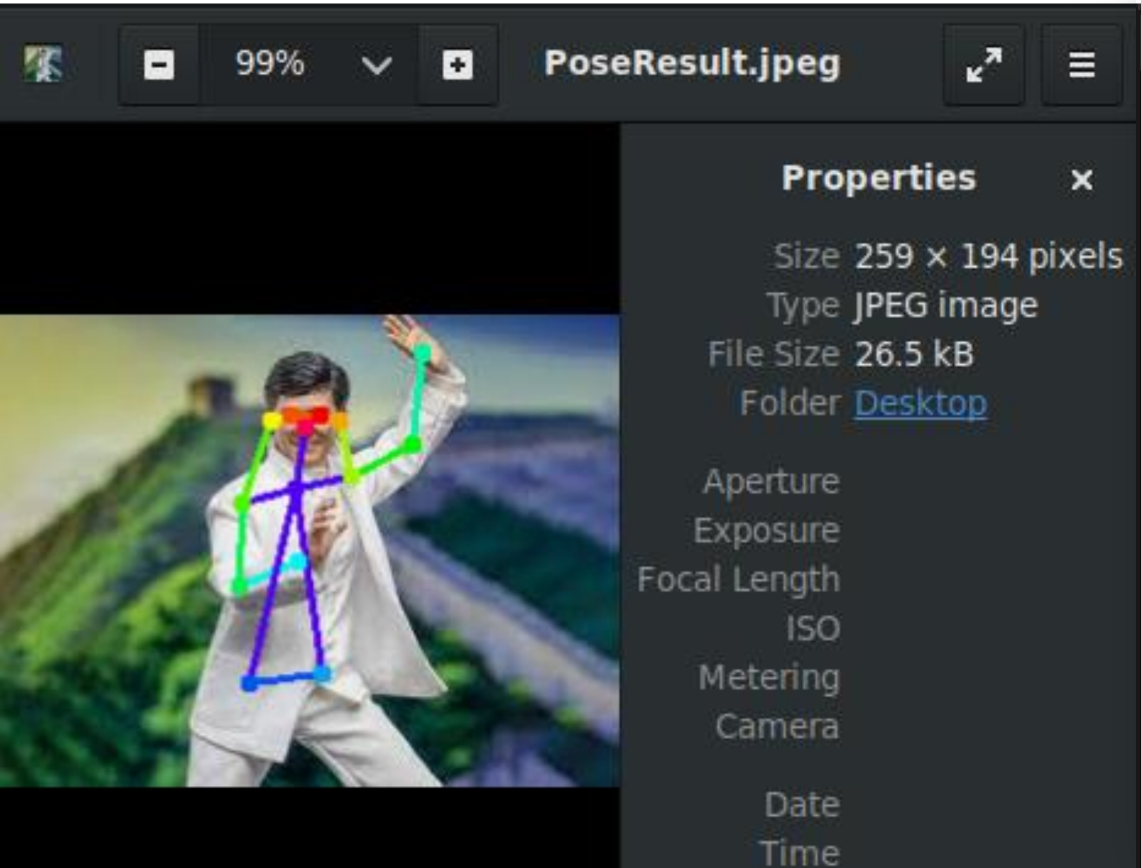
- **RMPE: Regional Multi-Person Pose Estimation（2017）AlphaPose**

RMPE can facilitate pose estimation in the presence of inaccurate human bounding boxes. The approach follows the two-step framework, first detects human bounding boxes and then estimating the pose within each box independently. (video link: https://www.youtube.com/watch?v=Z2WPd59pRi8&ab_channel=haoshufang)



Pose estimation --- Image

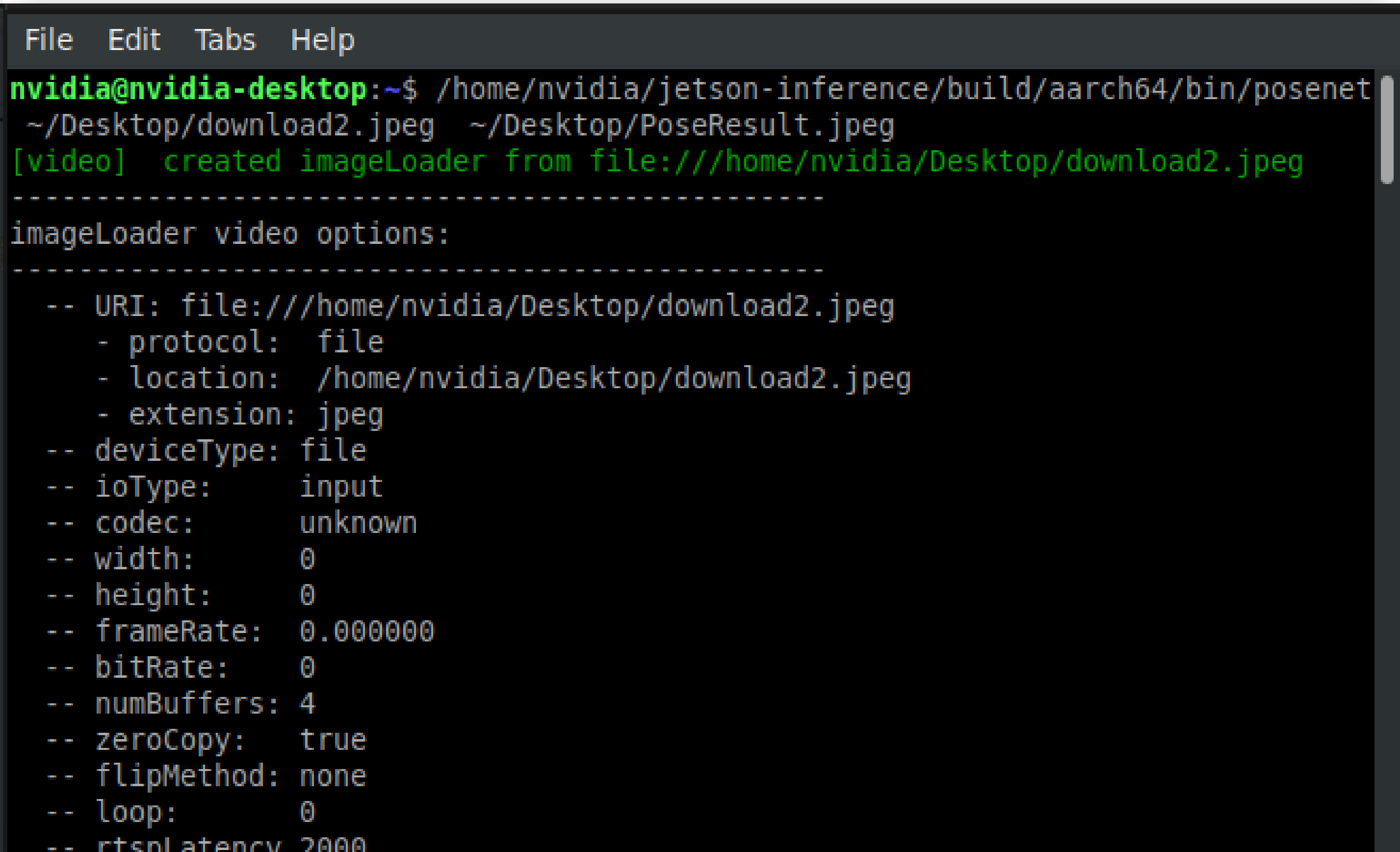
- $ /home/nvidia/jetson-inference/build/arrch64/bin/posenet *YourImagePath ResultSavePath*



Size 259 × 194 pixels
Type JPEG image
File Size 26.5 kB
Folder Desktop

Aperture
Exposure
Focal Length
ISO
Metering
Camera

Date
Time



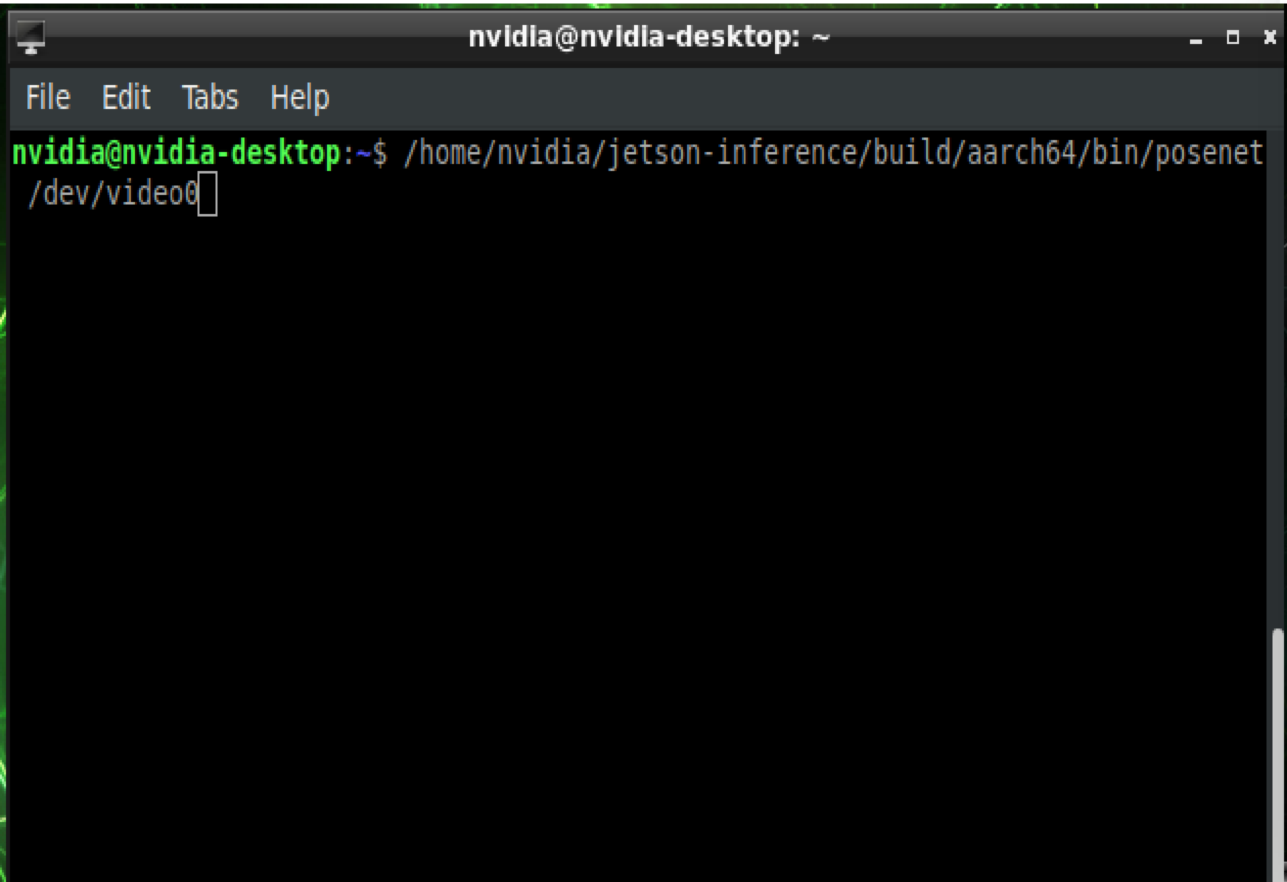
```
nvidia@nvidia-desktop:~$ /home/nvidia/jetson-inference/build/aarch64/bin/posenet
 ~/Desktop/download2.jpeg  ~/Desktop/PoseResult.jpeg
[video]  created imageLoader from file:///home/nvidia/Desktop/download2.jpeg
------------------------------------------------
imageLoader video options:
------------------------------------------------
  -- URI: file:///home/nvidia/Desktop/download2.jpeg
     - protocol:  file
     - location:  /home/nvidia/Desktop/download2.jpeg
     - extension: jpeg
  -- deviceType: file
  -- ioType:     input
  -- codec:      unknown
  -- width:      0
  -- height:     0
  -- frameRate:  0.000000
  -- bitRate:    0
  -- numBuffers: 4
  -- zeroCopy:   true
  -- flipMethod: none
  -- loop:       0
  -- rtspLatency 2000
```

Pose estimation --- Video

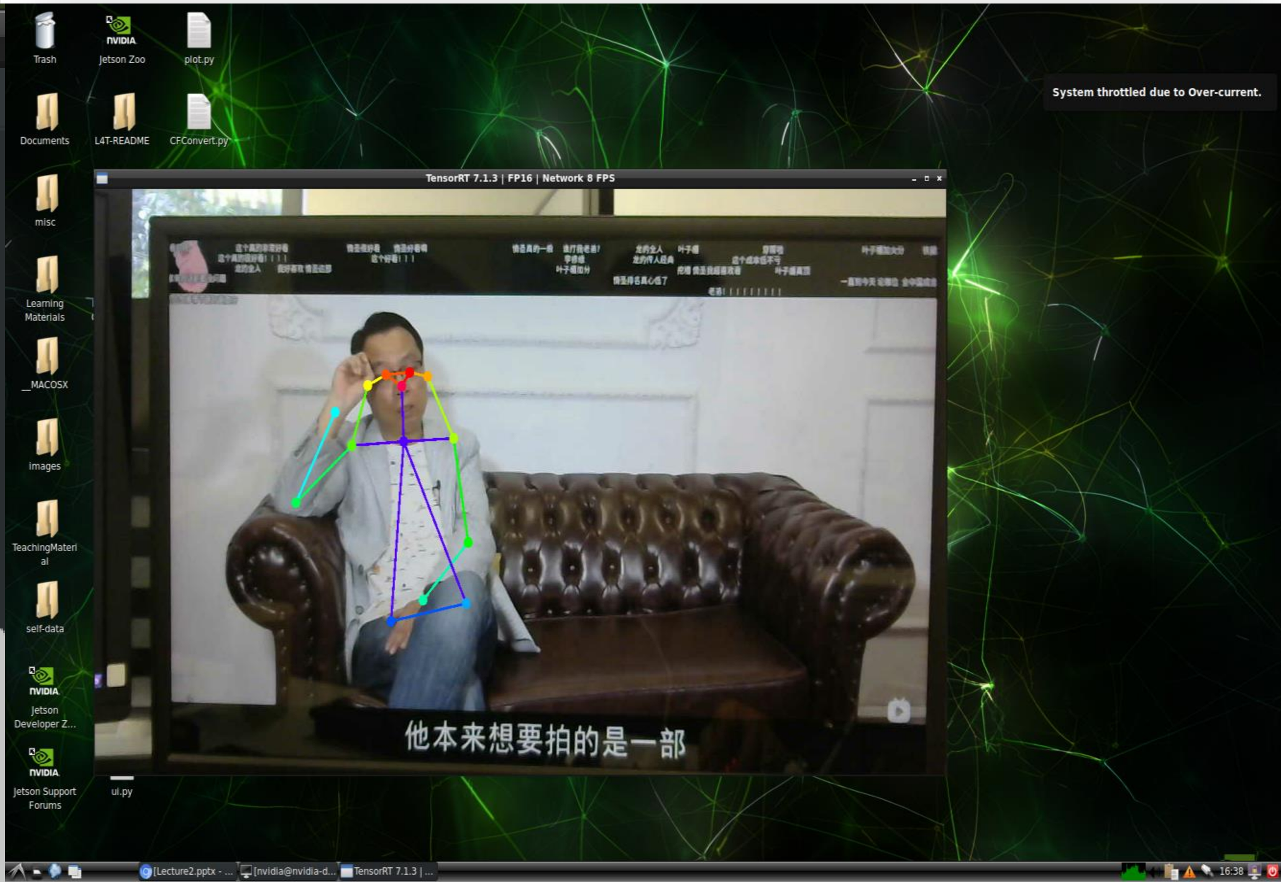
- $ /home/nvidia/jetson-inference/build/aarch64/bin/posenet /dev/video0

OR

- $ cd /home/nvidia/jetson-inference/build/aarch64/bin
- $ posenet /dev/video0



```
nvidia@nvidia-desktop:~$ /home/nvidia/jetson-inference/build/aarch64/bin/posenet
 /dev/video0
```



How to get the coordinate of keypoints?

Use the pretrained model Pose-ResNet18-Body

Input an image, we can get the object bbox, keypoints and links as output.

```
# load the pose estimation model
net = jetson.inference.poseNet(opt.network, sys.argv, opt.threshold)
# perform pose estimation (with overlay)
poses = net.Process(img, overlay=opt.overlay)
for pose in poses:
    print(pose.Keypoints)  #print keypoints id, x, y. (x,y) is the coordinate
```

Use PoseNet to get the keypoints and an angle

If we want to detect the bending angle of the right arm of this lady, how should we implement it in code?

We have 18 keypoints:["nose", "left_eye", "right_eye", "left_ear", "right_ear", "left_shoulder", "right_shoulder", "left_elbow", "right_elbow", "left_wrist", "right_wrist", "left_hip", "right_hip", "left_knee", "right_knee", "left_ankle", "right_ankle", "neck"]

Use PoseNet to get the keypoints and an angle

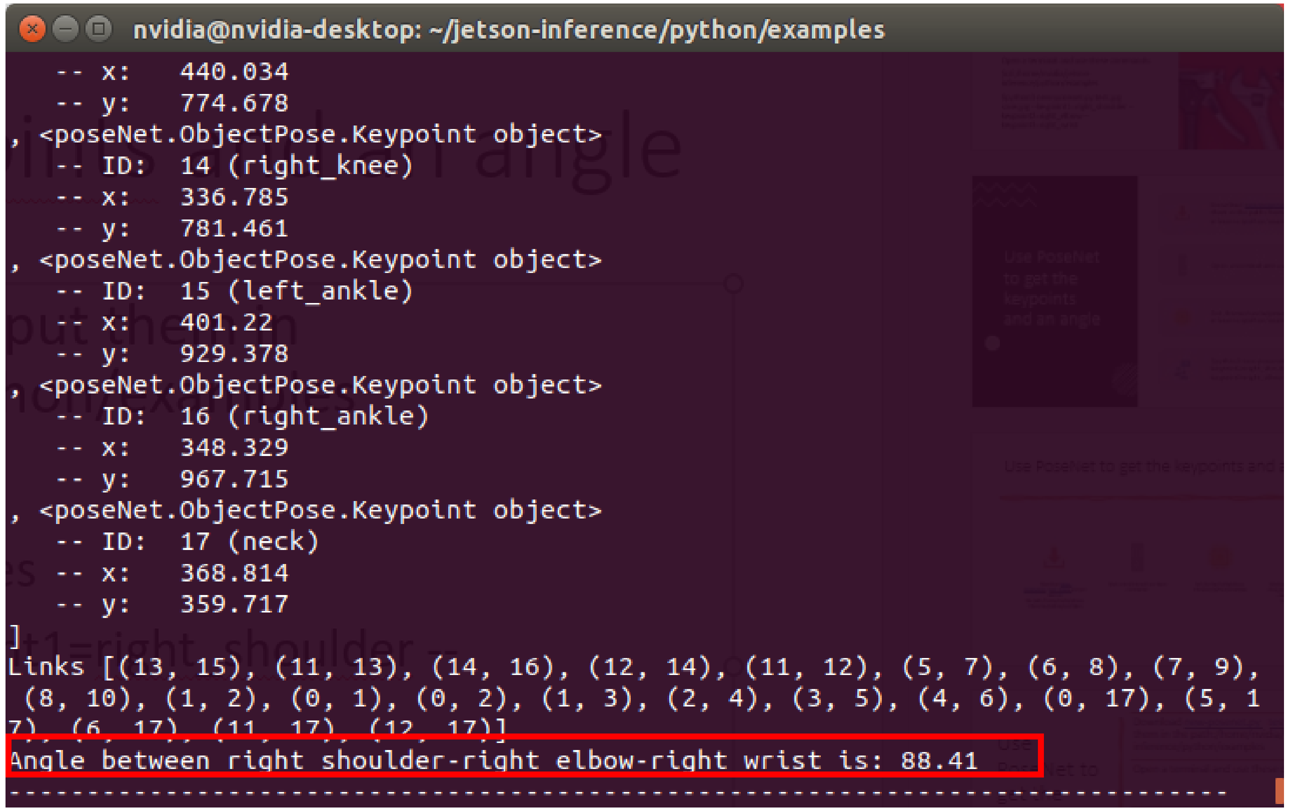
Download new-posenet.py   test_image and put them in the path: /home/nvidia/jetson-inference/python/examples

Open a terminal and use these commands:

$cd /home/nvidia/jetson-inference/python/examples

$python3 new-posenet.py test.jpg save.jpg

In this case, we calculate the angle right_shoulder-right_elbow-right_wrist



nvidia@nvidia-desktop: ~/jetson-inference/python/examples

```
   -- x:    440.034
   -- y:    774.678
, <poseNet.ObjectPose.Keypoint object>
   -- ID:   14 (right_knee)
   -- x:    336.785
   -- y:    781.461
, <poseNet.ObjectPose.Keypoint object>
   -- ID:   15 (left_ankle)
   -- x:    401.22
   -- y:    929.378
, <poseNet.ObjectPose.Keypoint object>
   -- ID:   16 (right_ankle)
   -- x:    348.329
   -- y:    967.715
, <poseNet.ObjectPose.Keypoint object>
   -- ID:   17 (neck)
   -- x:    368.814
   -- y:    359.717
]
Links [(13, 15), (11, 13), (14, 16), (12, 14), (11, 12), (5, 7), (6, 8), (7, 9),
 (8, 10), (1, 2), (0, 1), (0, 2), (1, 3), (2, 4), (3, 5), (4, 6), (0, 17), (5, 1
7), (6, 17), (11, 17), (12, 17)]
Angle between right shoulder-right elbow-right wrist is: 88.41
-----------------------------------------------------------------------------
```

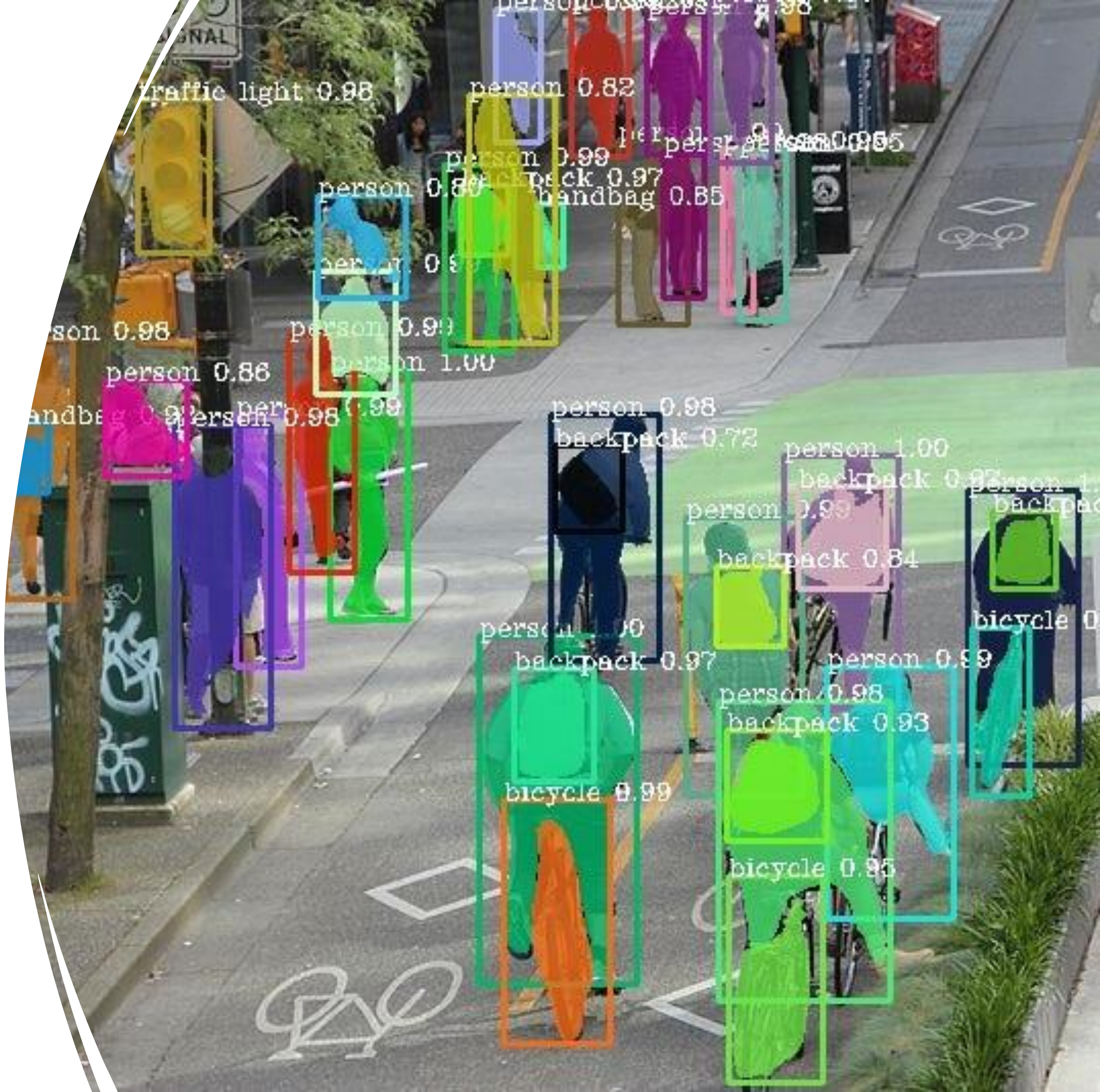
How can we utilise the keypoint to do the calculation?

```
def calculate_angle(id1, id2, id3):
    point1_x = np.array(pose.Keypoints[id1].x)
    point1_y = np.array(pose.Keypoints[id1].y)
    point2_x = np.array(pose.Keypoints[id2].x)
    point2_y = np.array(pose.Keypoints[id2].y)
    point3_x = np.array(pose.Keypoints[id3].x)
    point3_y = np.array(pose.Keypoints[id3].y)
    v1 = (point1_x - point2_x, point1_y - point2_y)
    v2 = (point3_x - point2_x, point3_y - point2_y)
    cosine_angle = np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))
    angle = np.arccos(cosine_angle)
    return np.degrees(angle)
```

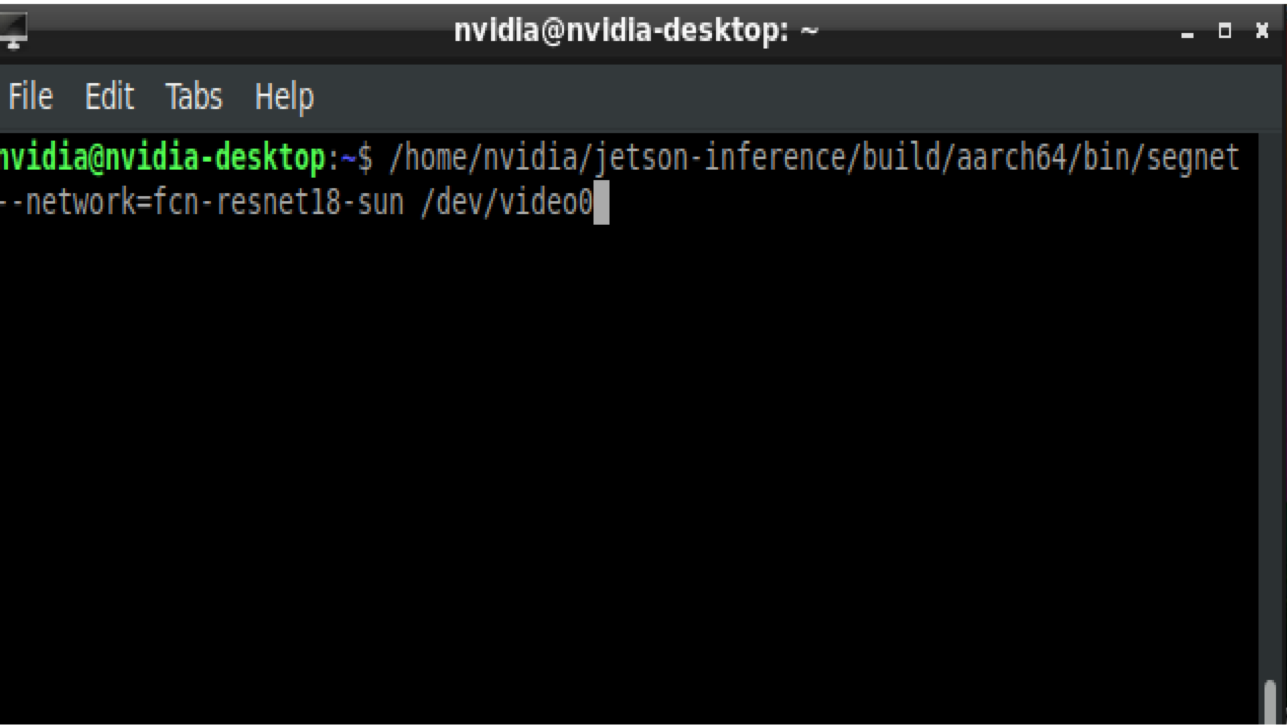
Calculate the angle among 3 keypoints:
Get three keypoints' coordinates, and calculate the two vectors:
$\theta = \cos^{-1} [(a. b) / (|a|\ |b|)]$
The python code is:

```
    cosine_angle = np.dot(v1, v2) /
     (np.linalg.norm(v1) * np.linalg.norm(v2))
    angle = np.arccos(cosine_angle)
    return np.degrees(angle)
```

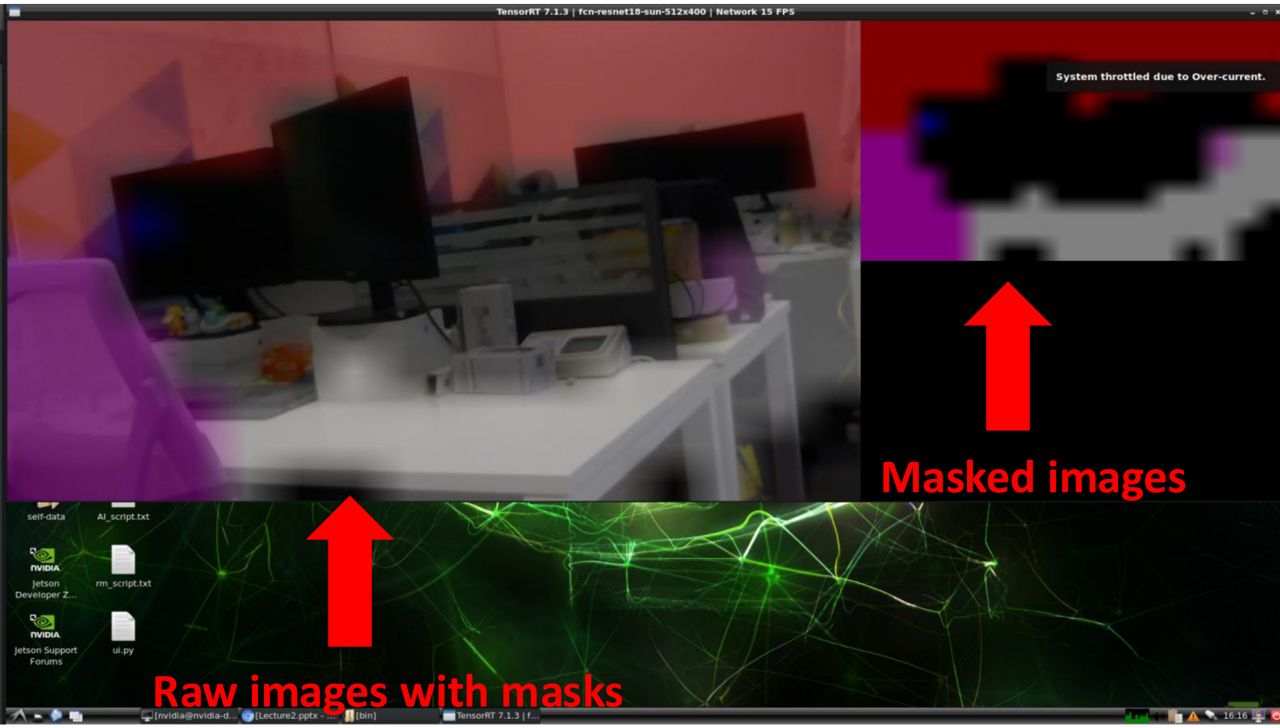
Object segmentation



Object segmentation --- Video



```
nvidia@nvidia-desktop: ~
File  Edit  Tabs  Help
nvidia@nvidia-desktop:~$ /home/nvidia/jetson-inference/build/aarch64/bin/segnet
--network=fcn-resnet18-sun /dev/video0
```


TensorRT 7.1.3 | fcn-resnet18-sun-512x400 | Network 15 FPS
System throttled due to Over-current.
Masked images
Raw images with masks

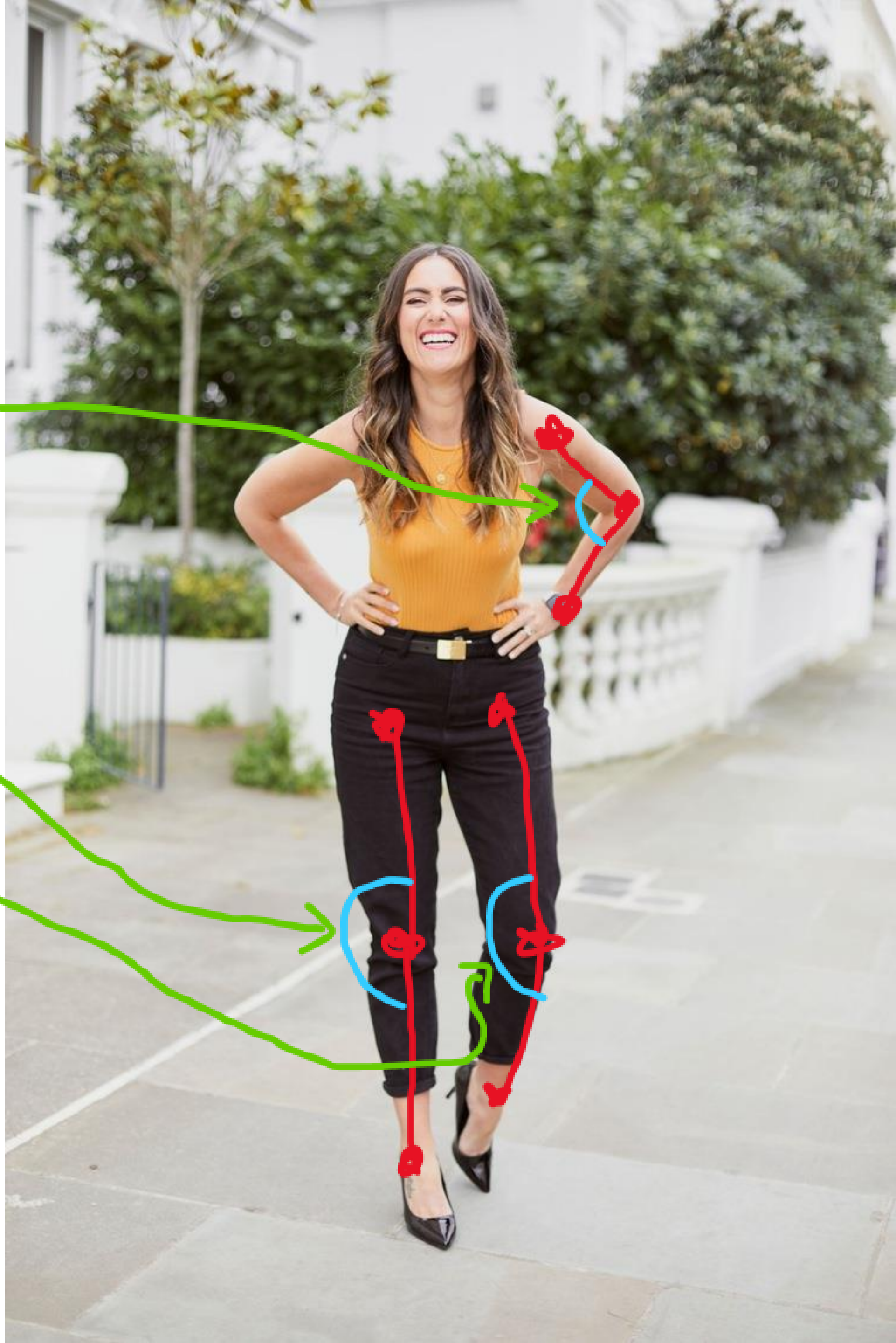
Download model and put it in /home/nvidia/jetson-inference/build/aarch64/bin/networks/
Then run the command in the terminal:
$ /home/nvidia/jetson-inference/build/aarch64/bin/segnet --network=fcn-resnet18-voc /dev/video0

Assignment 4

- Detect the angle of the left arm
- Detect the angle of the right leg
- Detect the angle of the left leg
- Show the angle value on the image

Input image: test_image

Deadline: the end of the today's course (9:30pm)
Assessment in class by TA

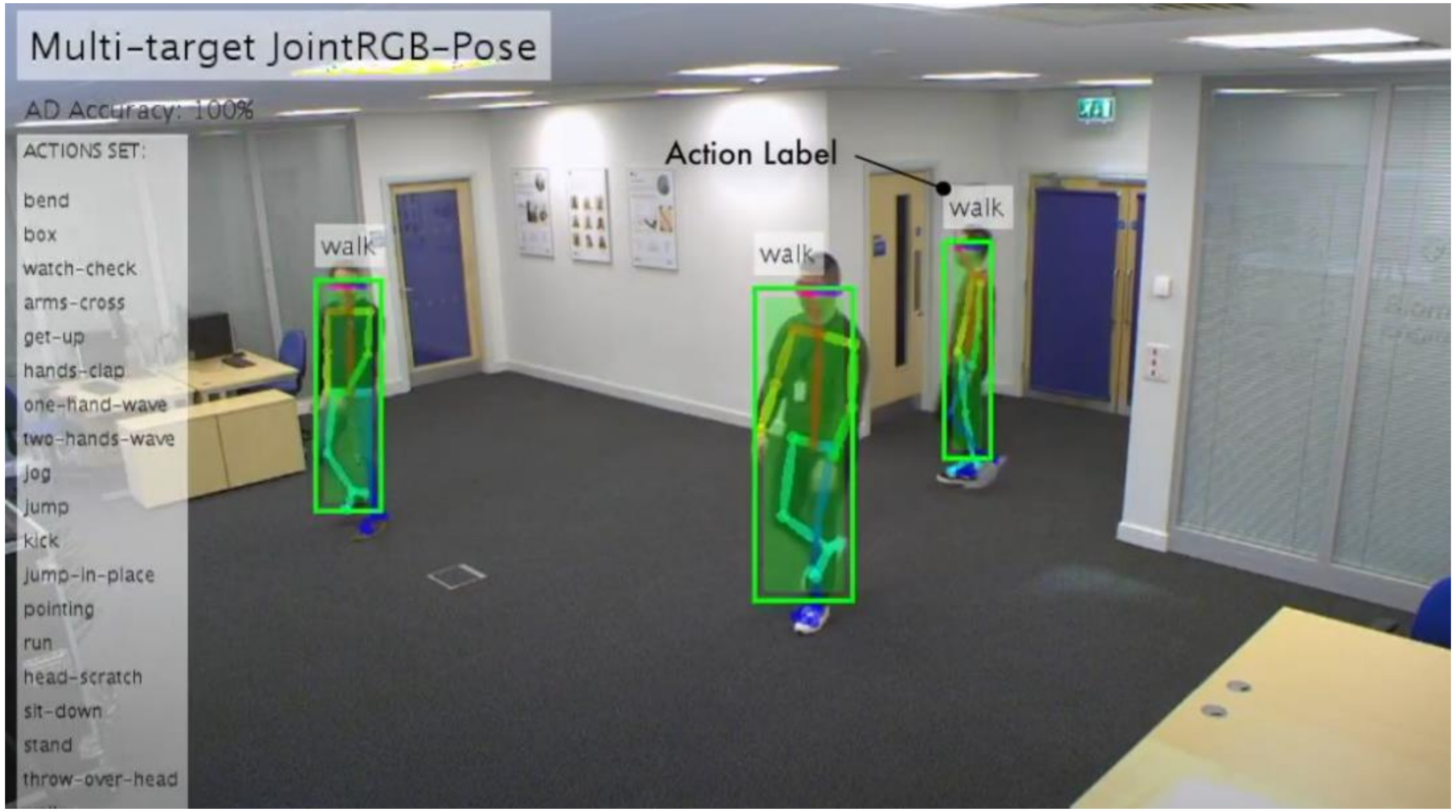


Action Recognition

Action Recognition in Videos

Action Recognition in Videos is a task in computer vision where the goal is to **identify and categorize human actions performed in a video sequence**. The task involves analyzing the spatiotemporal dynamics of the actions and mapping them to a predefined set of action classes, such as running, jumping, or swimming.

Demo of Action Recognition



https://youtu.be/7_mcWCB76Ps?list=PL18uP4AzeyVuqH7dRNmyLGR1USnSgfliV

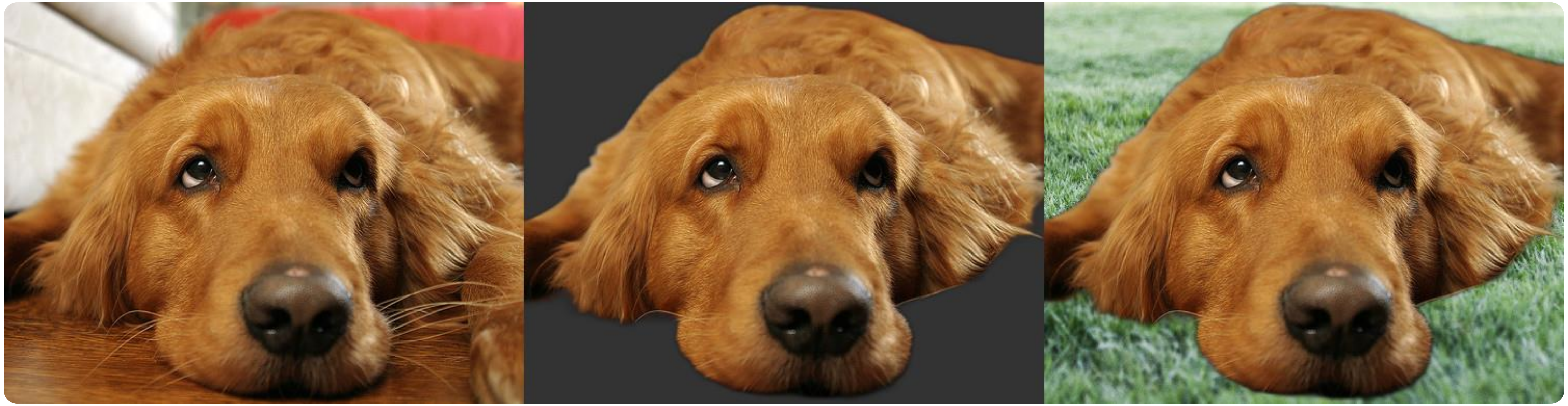
Run ActionNet

- Download the demo video and put it in the path:
- $ cd Jeston-inference/python/examples
- $ python3 actionnet.py demo.mp4 result.mp4

Run ActionNet-Camera

- $ cd Jeston-inference/python/examples
- $ python3 actionnet.py /dev/video0

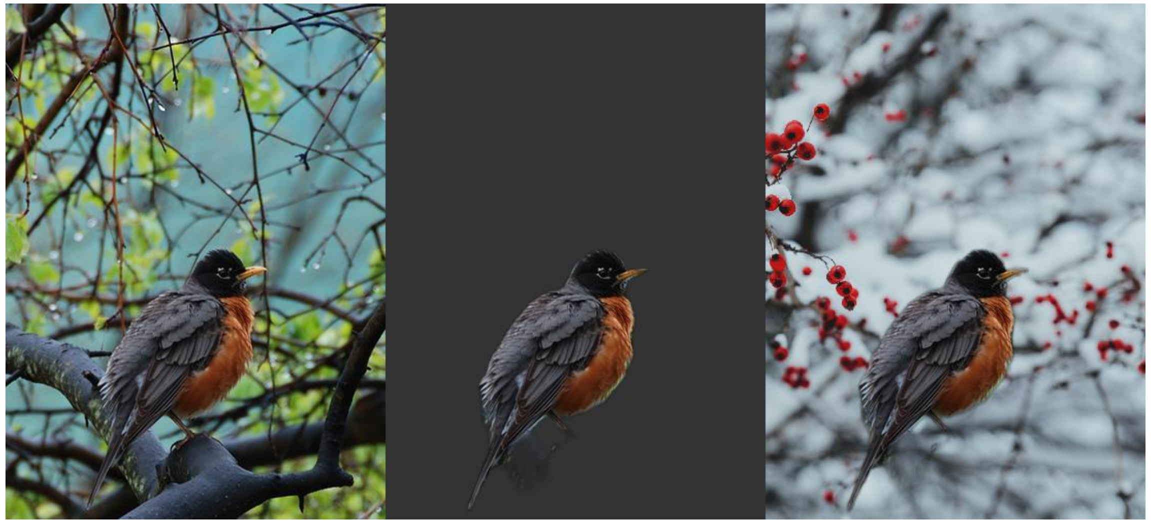
Background Removal



Background Removal

Background removal (aka background subtraction, or salient object detection) **generates a mask that segments the foreground from the background** of an image. It can be used to replace or blur backgrounds (similar to video conferencing applications), or it could aid in pre-processing for other vision DNN's like object detection/tracking or motion detection.

Run BackgroundNet

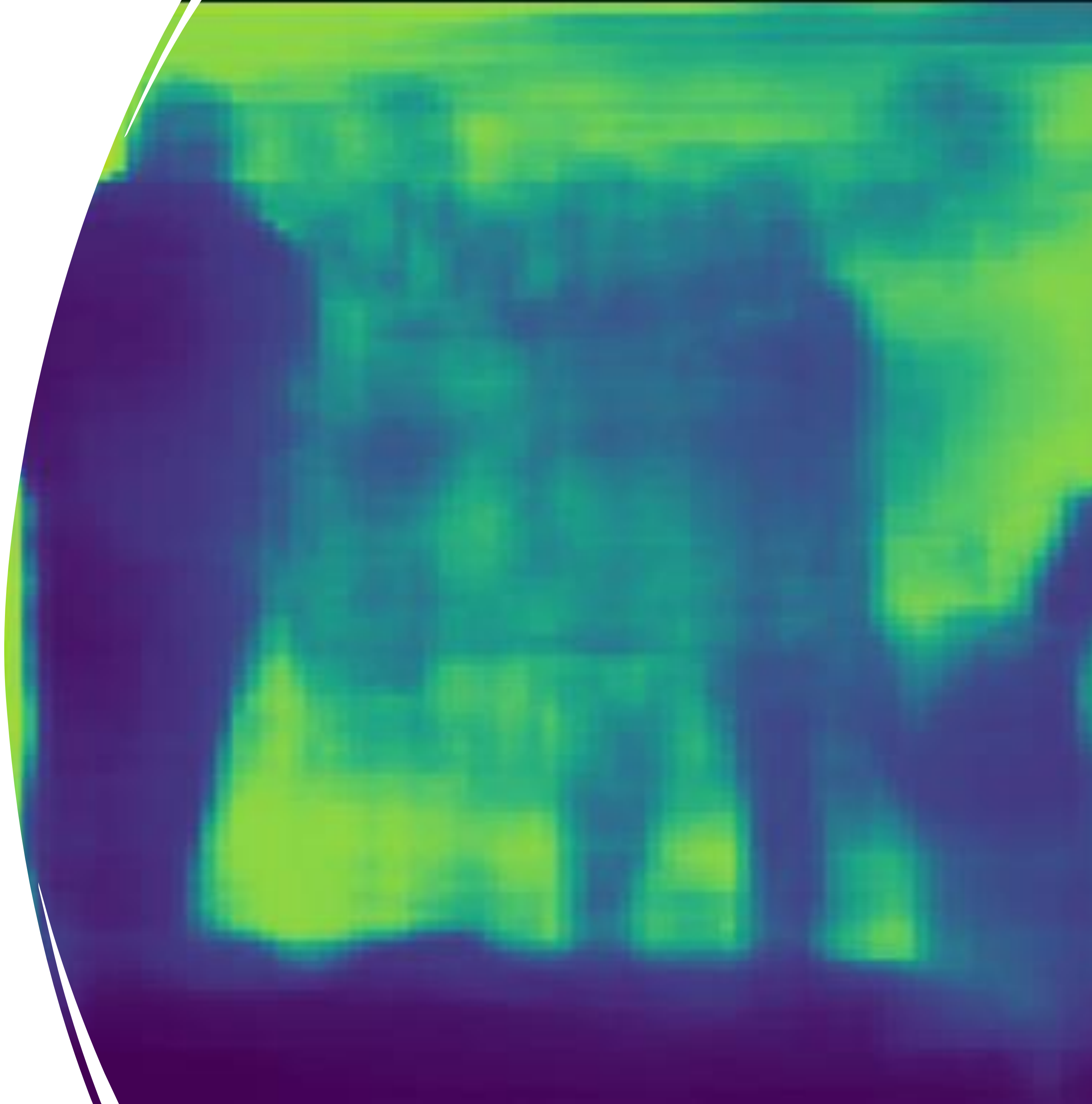


- Download the demo image and put it in the path:
- $ cd Jeston-inference/python/examples
- $ python3 backgroundnet.py images/bird_0.jpg images/test/bird_mask.png # remove the background (with alpha)
- $ python3 backgroundnet.py --replace=images/snow.jpg images/bird_0.jpg images/test/bird_replace.jpg # replace the background

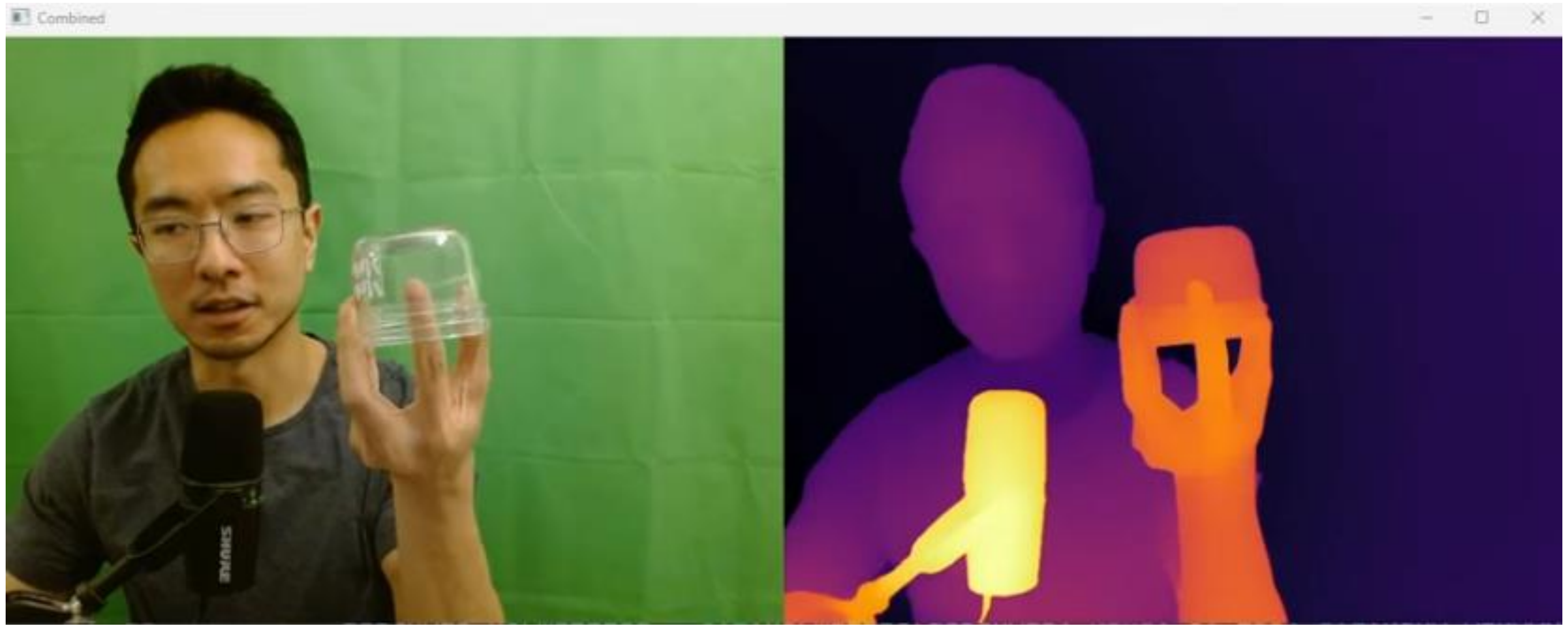
Run BackgroundNet-Camera

- $ cd Jeston-inference/python/examples
- $ python3 backgroundnet /dev/video0 # remove the background
- $ python3 backgroundnet --replace=images/coral.jpg /dev/video0
  # replace the background

Monocular Depth Estimation



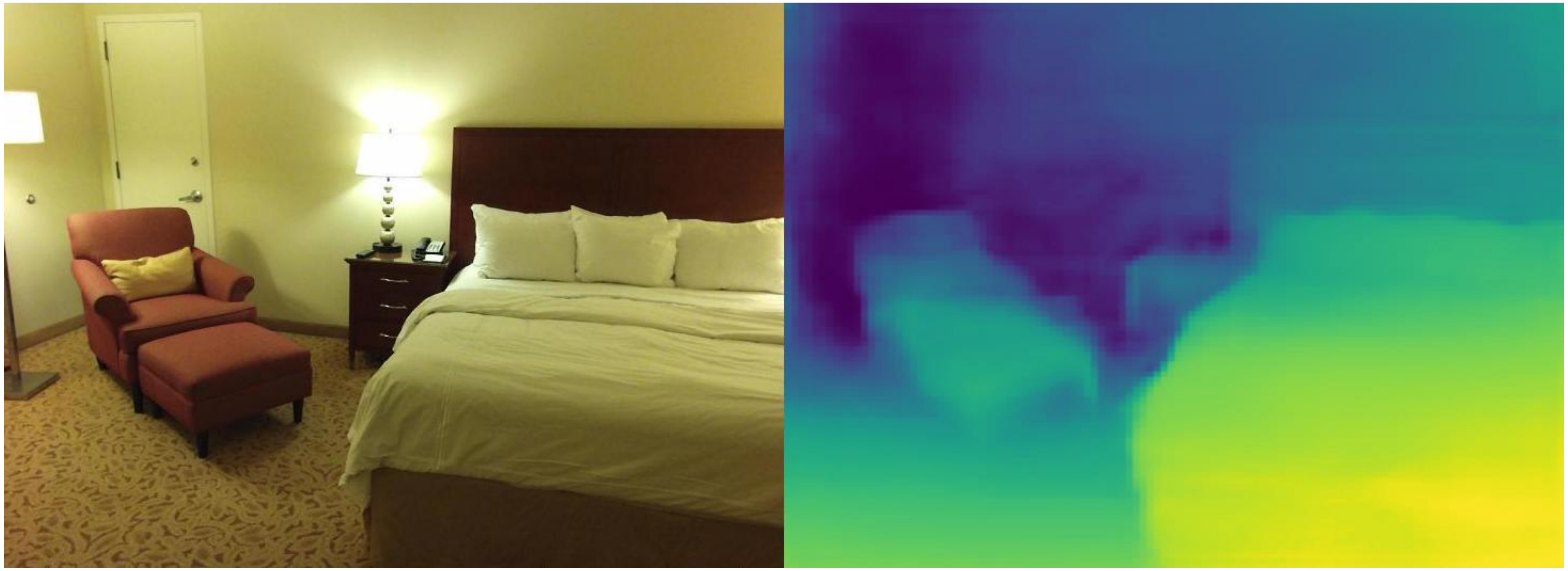
Monocular Depth Estimation is the task of estimating **the depth value (distance relative to the camera) of each pixel** given a single (monocular) RGB image. This challenging task is a key prerequisite for determining scene understanding for applications such as 3D scene reconstruction, autonomous driving, and AR.



Real time Depth Anything V2 demo

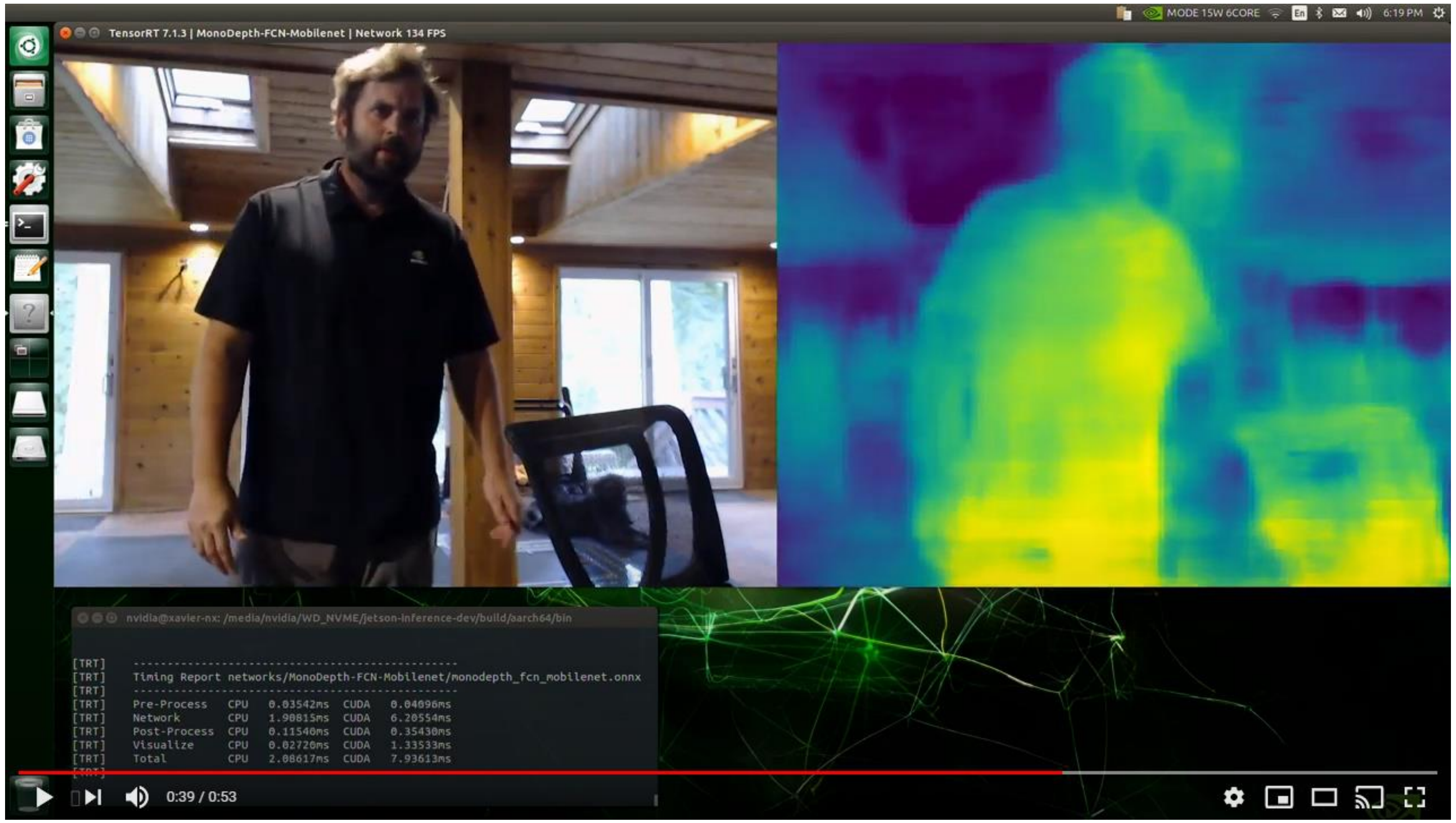
Mono Depth on Images

- $ cd Jeston-inference/python/examples
- $ python3 depthnet.py "images/room_*.jpg" images/test/depth_room_%i.jpg



Mono Depth from video

- $ cd Jeston-inference/python/examples
- $ python3 depthnet.py /dev/video0



```
nvidia@xavier-nx: /media/nvidia/WD_NVME/jetson-inference-dev/build/aarch64/bin

[TRT]    ----------------------------------------------
[TRT]    Timing Report networks/MonoDepth-FCN-Mobilenet/monodepth_fcn_mobilenet.onnx
[TRT]    ----------------------------------------------
[TRT]    Pre-Process   CPU   0.03542ms  CUDA   0.04096ms
[TRT]    Network       CPU   1.90815ms  CUDA   6.20554ms
[TRT]    Post-Process  CPU   0.11540ms  CUDA   0.35430ms
[TRT]    Visualize     CPU   0.02720ms  CUDA   1.33533ms
[TRT]    Total         CPU   2.08617ms  CUDA   7.93613ms
```