
ROS

ROS OpenCV integration for image processing

OpenCV

OpenCV

- ชื่อเต็มคือ Open Source Computer Vision Library
- เป็น open source
  - Computer vision
  - Machine learning
- โดยมีการจัดให้มี
  - โครงสร้างพื้นฐานทั่วไปของ computer vision
  - มีตัวช่วยสำหรับ accelerate machine perception

OpenCV

- 2500 classic and state-of-the-art optimized algorithm for computer vision and machine learning
  - detect and recognize faces
  - identify objects
  - classify human actions in videos
  - track camera movements
  - track moving objects
  - extract 3D models of objects
  - produce 3D point clouds from stereo cameras
  - stitch images together to produce a high resolution image
  - find similar images from an image database
  - follow eye movements
  - etc.

OpenCV

- ภาษาที่ support
  - C++
  - Python
  - Java
  - MATLAB
- OS ที่ support
  - Windows
  - Linux
  - Android
  - Mac OS

OpenCV

REF: https://en.wikipedia.org/wiki/CUDA

REF: https://en.wikipedia.org/wiki/OpenCL

Install python OpenCV

```
$ sudo apt update
```

```
$ sudo apt install python3-pip
```

```
$ pip3 install opencv-python
```

OpenCV in python

```
$ cd ~/tutorial_ws/src/
```

```
$ catkin_create_pkg my_ros_opencv rospy std_msgs sensor_msgs
```

```
$ cd ..
```

```
$ catkin_make
```

```
$ rospack profile
```

```
$ cd src/my_ros_opencv/src/
```

```
$ gedit test_opencv.py
```

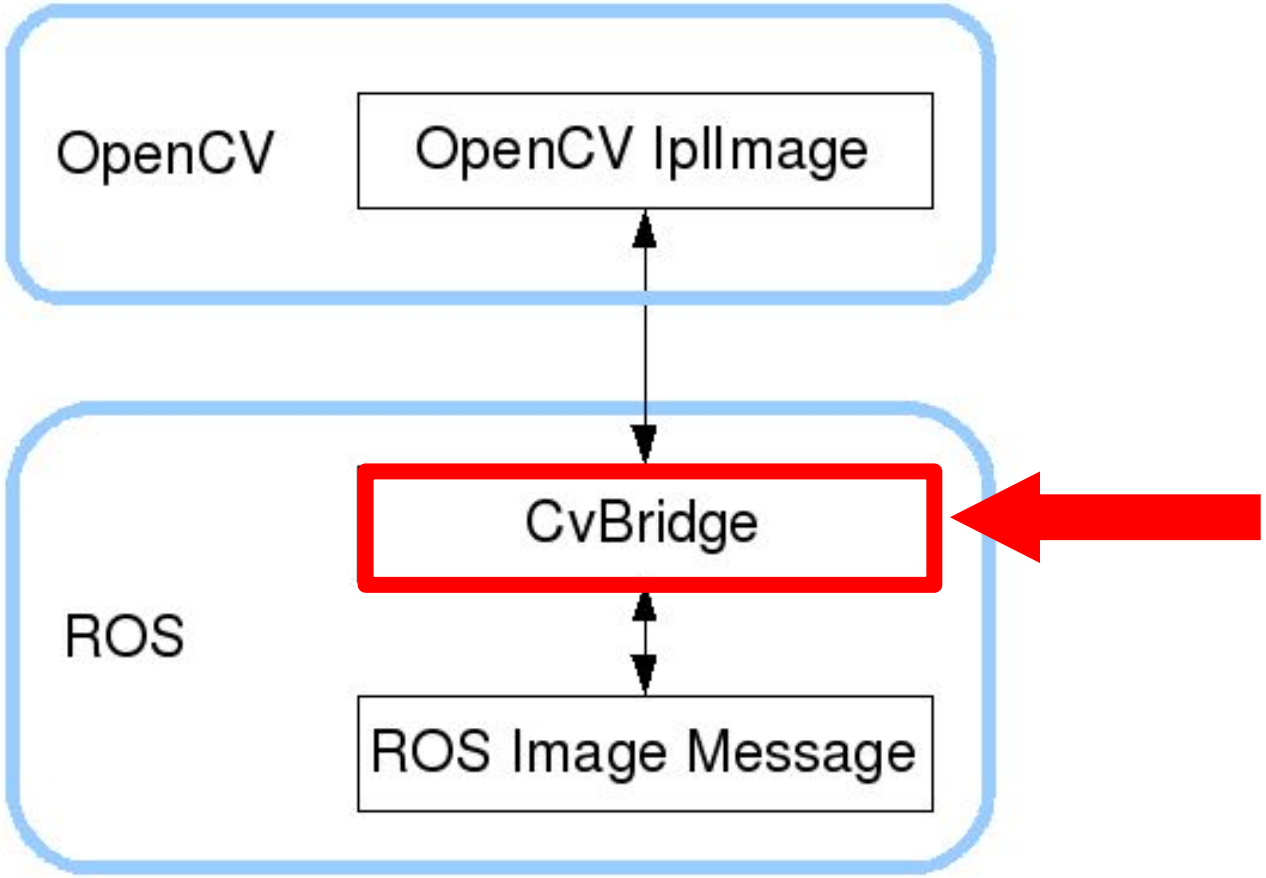
test_opencv.py

```
#!/usr/bin/env python3

import cv2
print(cv2.__file__)
```

```
$ python3 test_opencv.py
```

```
[redacted]:~/tutorial_ws/src/my_ros_opencv/src$ python3 test_opencv.py
/usr/local/lib/python3.8/dist-packages/cv2/__init__.py
```

Send data to ROS


OpenCV
OpenCV IplImage
CvBridge
ROS
ROS Image Message

```
$ cd ..
```

```
$ mkdir vdo
```

```
$ cd vdo
```

```
$ wget https://github.com/RobotCitizens/ros-course-material/raw/main/file/promote.mp4
```

```
$ cd ../src
```

```
$ gedit send_to_ros.py
```

send_to_ros.py

send_to_ros.py

```
#!/usr/bin/env python3
import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
import rospkg

rospack = rospkg.RosPack()
cap = cv2.VideoCapture(rospack.get_path("my_ros_opencv")+"/vdo/promote.mp4")
bridge = CvBridge()
rospy.init_node("send_image_to_ros", anonymous=True)
pub = rospy.Publisher("test_image",Image,queue_size=10)
rate = rospy.Rate(30)
while not rospy.is_shutdown():
    ret, frame = cap.read()
    if not ret:
        break
    pub.publish(bridge.cv2_to_imgmsg(frame,"bgr8"))
    rate.sleep()
```

```
$ chmod +x send_to_ros.py
```

RUN

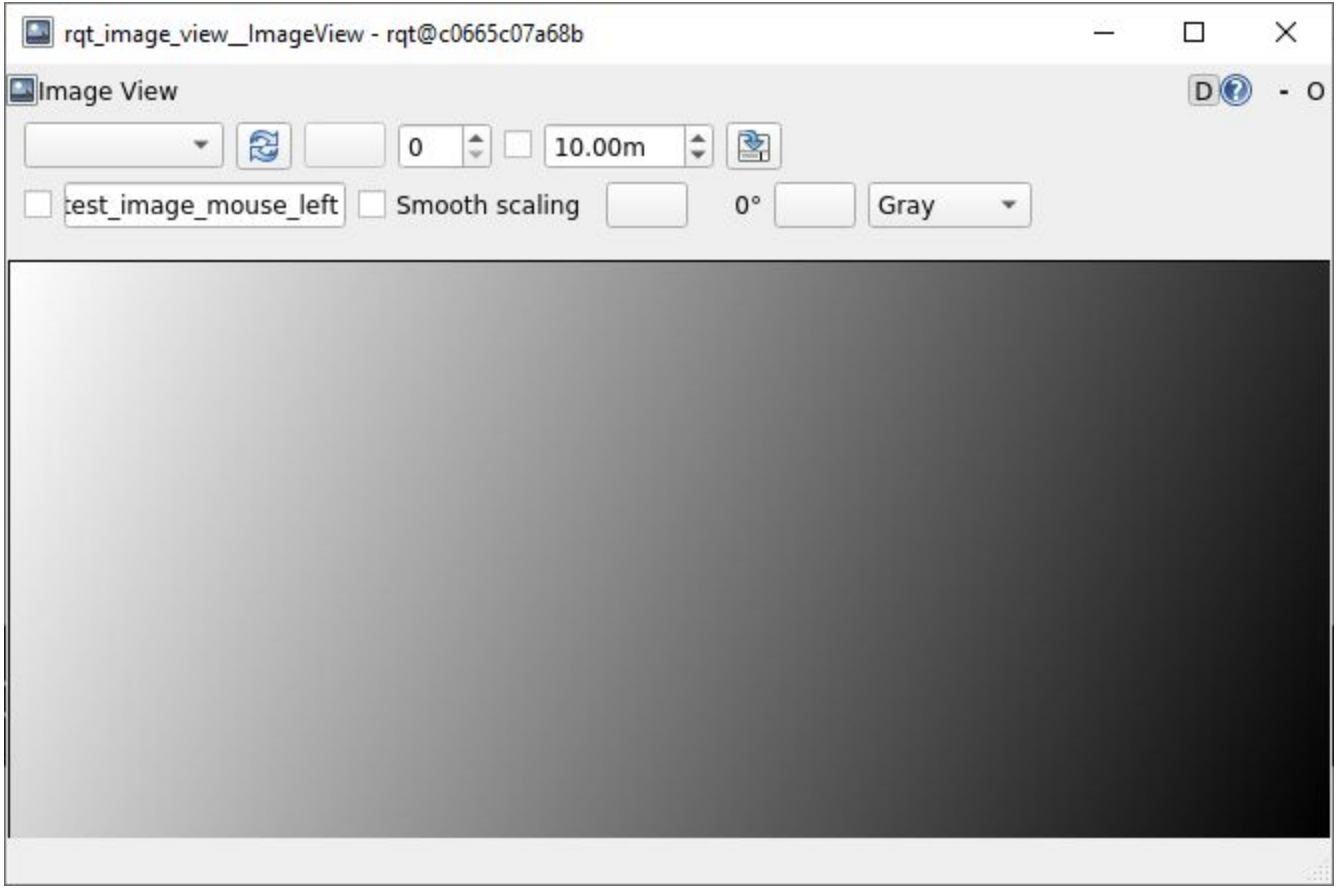
เปิด terminal

$ roscore

```
[redacted]:~# roscore
... logging to /root/.ros/log/a4938efa-7c5b-11ec-b410-0242ac110002/roslaunch-c0665c07a68b-2984.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://[redacted]:39823/
ros_comm version 1.15.13


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.13

NODES

auto-starting new master
process[master]: started with pid [3008]
ROS_MASTER_URI=http://[redacted]:11311/

setting /run_id to [redacted]
process[rosout-1]: started with pid [3028]
started core service [/rosout]
```

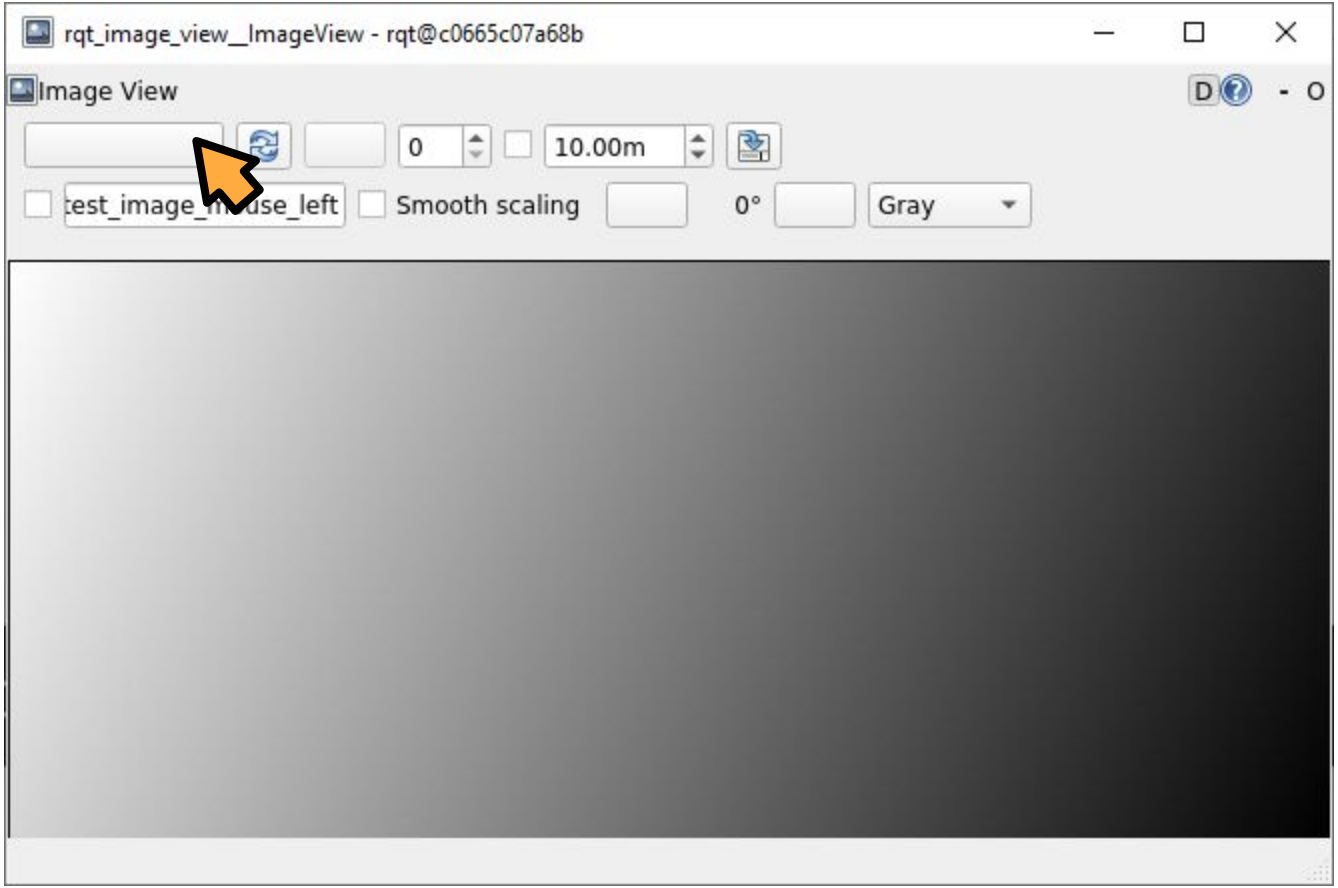
เปิดหน้าต่างใหม่

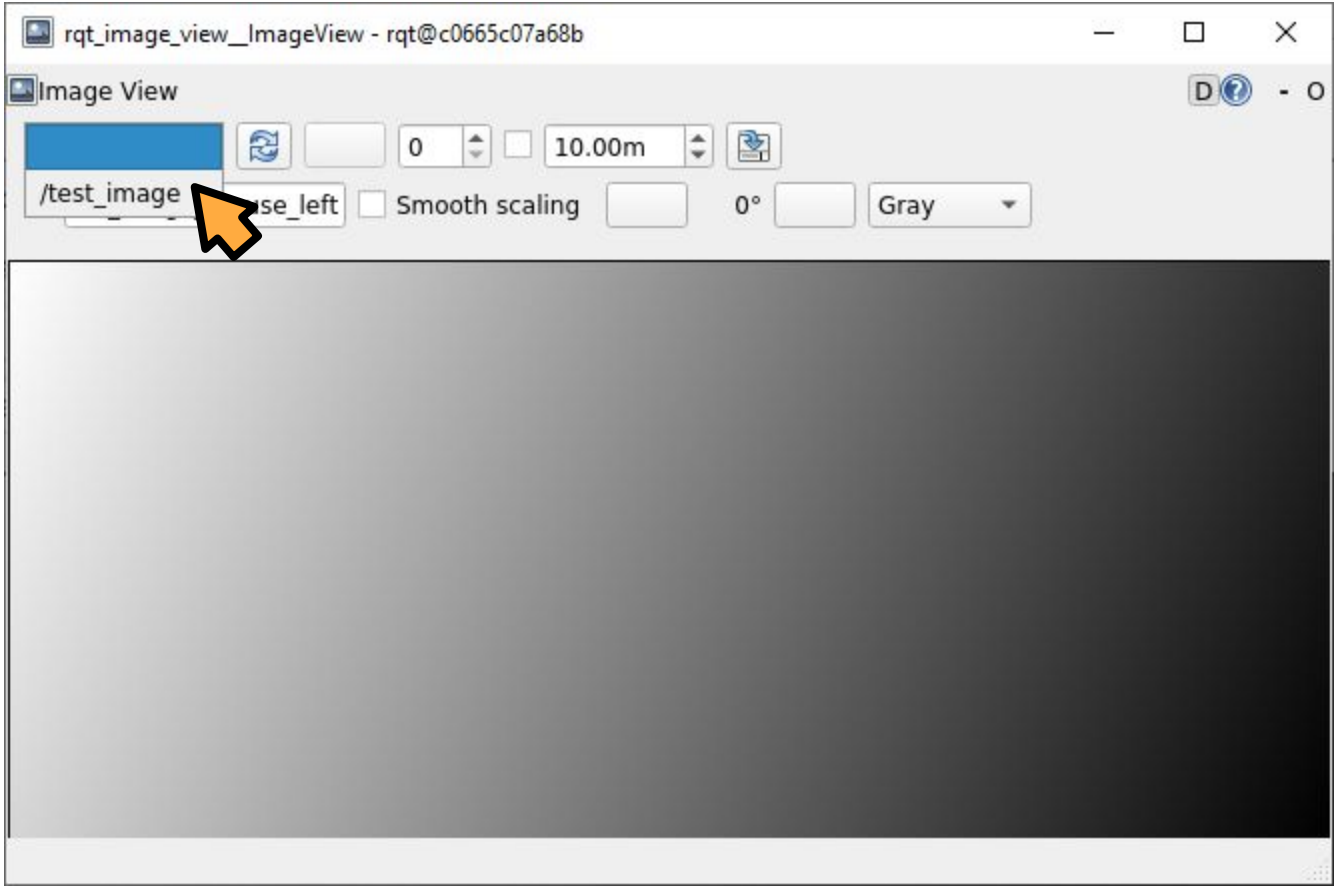
```
$ rqt_image_view
```

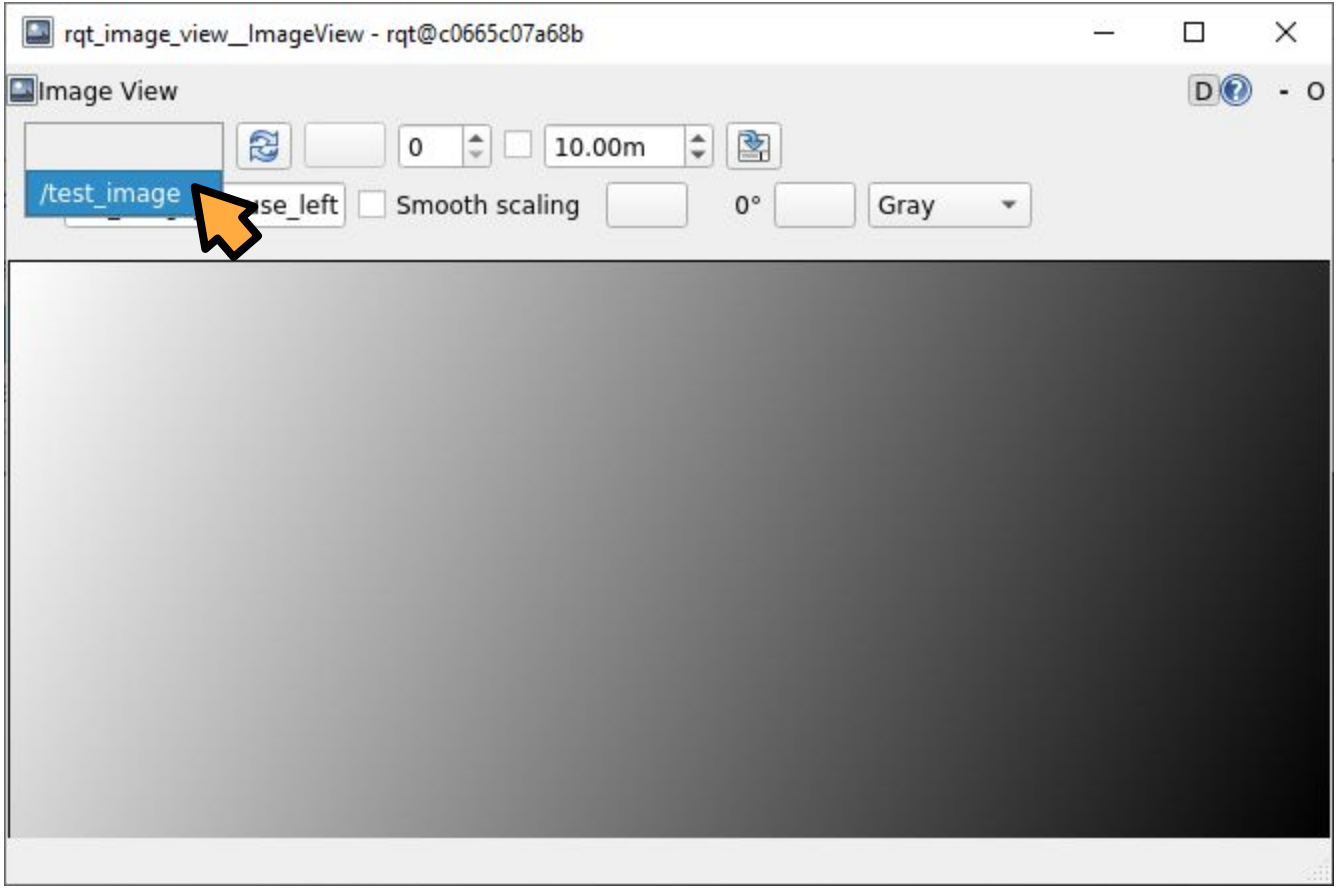

rqt_image_view__ImageView - rqt@c0665c07a68b
Image View
0
10.00m
test_image_mouse_left
Smooth scaling
0°
Gray

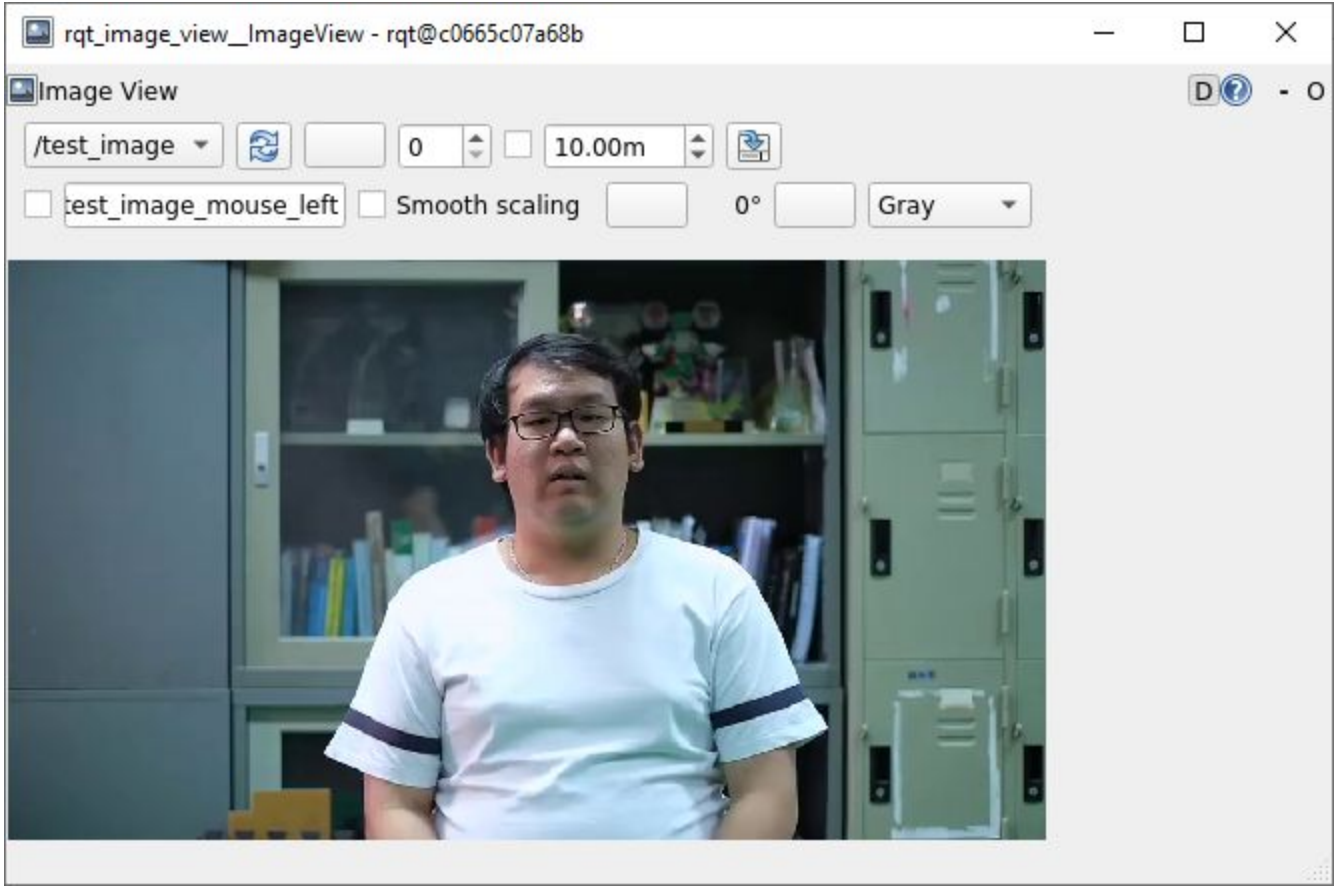
เปิดหน้าต่างใหม่

```
$ rosrun my_ros_opencv send_to_ros.py
```


rqt_image_view__ImageView - rqt@c0665c07a68b
Image View
0
10.00m
Smooth scaling
0°
Gray


rqt_image_view__ImageView - rqt@c0665c07a68b
Image View
0
10.00m
/test_image
Smooth scaling
0°
Gray


rqt_image_view__ImageView - rqt@c0665c07a68b
Image View
0
10.00m
/test_image
Smooth scaling
0°
Gray


rqt_image_view__ImageView - rqt@c0665c07a68b
Image View
/test_image
0
10.00m
test_image_mouse_left
Smooth scaling
0°
Gray

```
$ gedit my_face_detect_opencv_ros.py
```

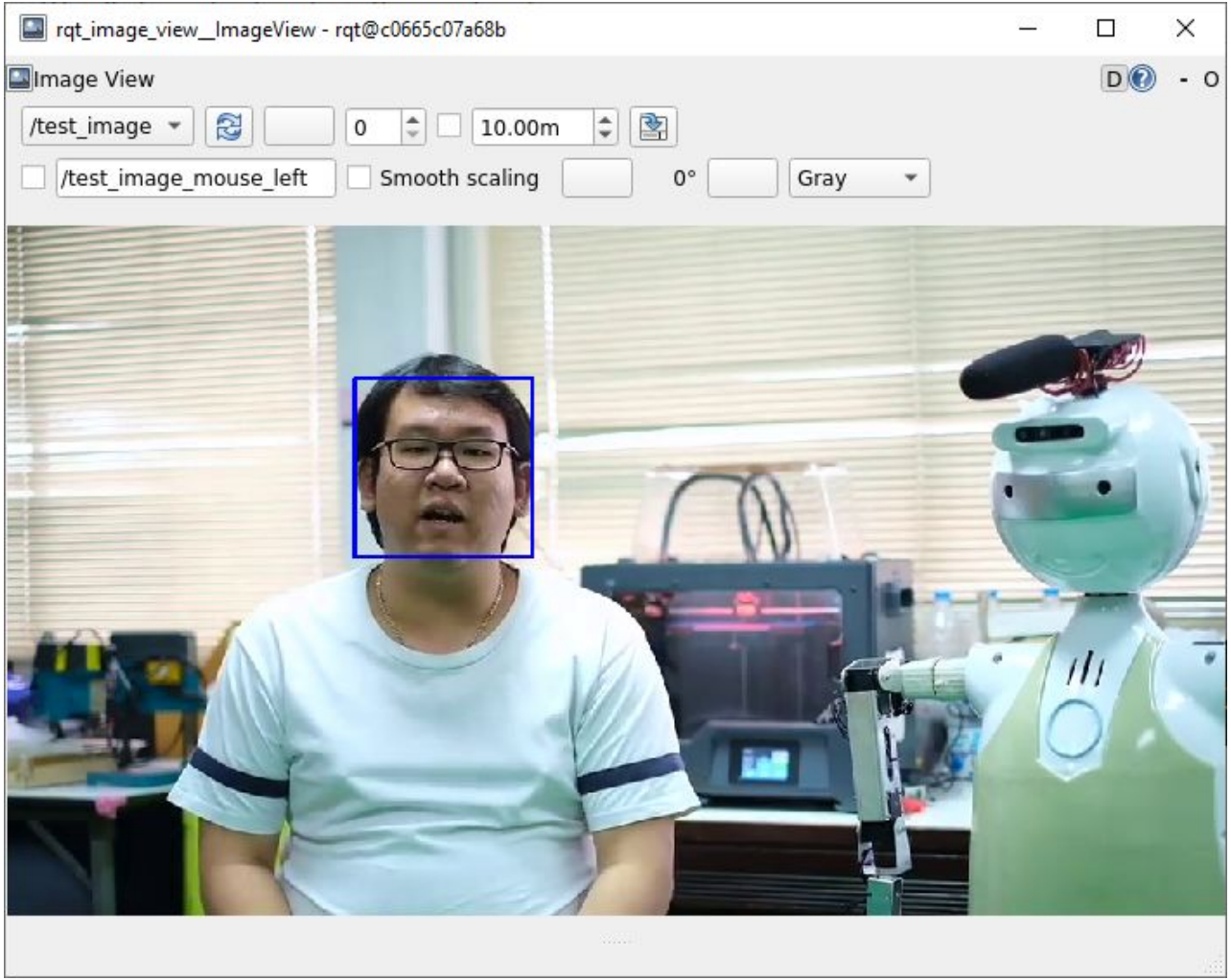
my_face_detect_opencv_ros.py

```python
#!/usr/bin/env python3
import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
import rospkg
```

```
rospack = rospkg.RosPack()
cap = cv2.VideoCapture(rospack.get_path("my_ros_opencv")+"/vdo/promote.mp4")
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
bridge = CvBridge()
rospy.init_node("send_image_to_ros", anonymous=True)
pub = rospy.Publisher("test_image",Image,queue_size=10)
rate = rospy.Rate(30)
while not rospy.is_shutdown():
        ret, frame = cap.read()
        if not ret:
                break
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 4)
        for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        pub.publish(bridge.cv2_to_imgmsg(frame,"bgr8"))
        rate.sleep()
```

```
$ chmod +x my_face_detect_opencv_ros.py
```

```
$ rosrun my_ros_opencv my_face_detect_opencv_ros.py
```


rqt_image_view__ImageView - rqt@c0665c07a68b
Image View
/test_image
0
10.00m
/test_image_mouse_left
Smooth scaling
0°
Gray

Receive data from ROS

```
$ gedit get_image.py
```

get_image.py

```python
#!/usr/bin/env python3

import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image

bridge = CvBridge()

def show_image(data):
    frame = bridge.imgmsg_to_cv2(data)
    cv2.imshow("frame", frame)
    cv2.waitKey(1)

rospy.init_node("get_image_from_ros", anonymous=True)
sub = rospy.Subscriber("test_image",Image,show_image)
rospy.spin()
```

```
$ chmod +x get_image.py
```

RUN

เปิด terminal

$ roscore

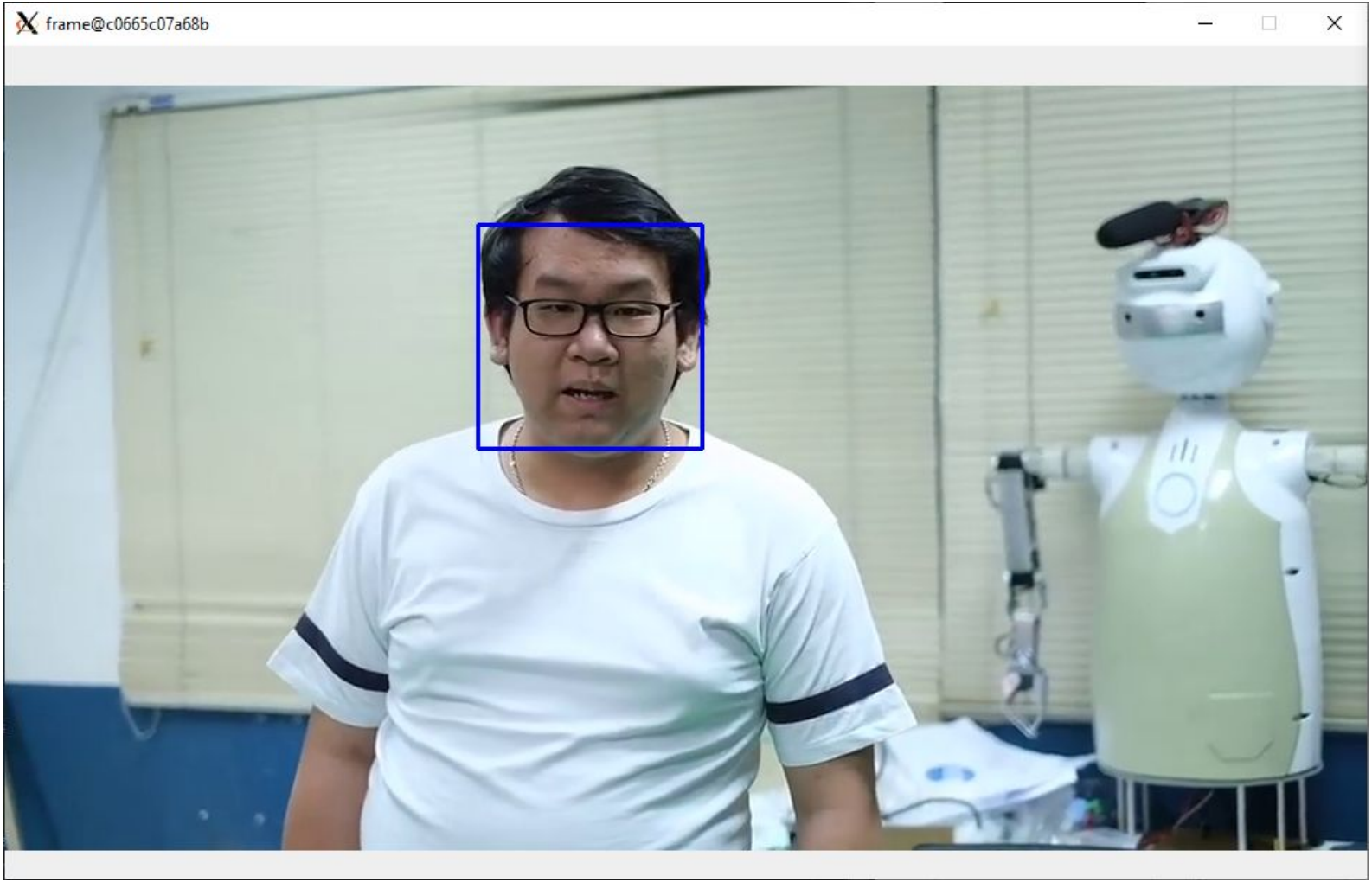
```
[redacted]:~# roscore
... logging to /root/.ros/log/a4938efa-7c5b-11ec-b410-0242ac110002/roslaunch-c0665c07a68b-2984.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://[redacted]:39823/
ros_comm version 1.15.13


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.13

NODES

auto-starting new master
process[master]: started with pid [3008]
ROS_MASTER_URI=http://[redacted]:11311/

setting /run_id to [redacted]
process[rosout-1]: started with pid [3028]
started core service [/rosout]
```

เปิดหน้าต่างใหม่

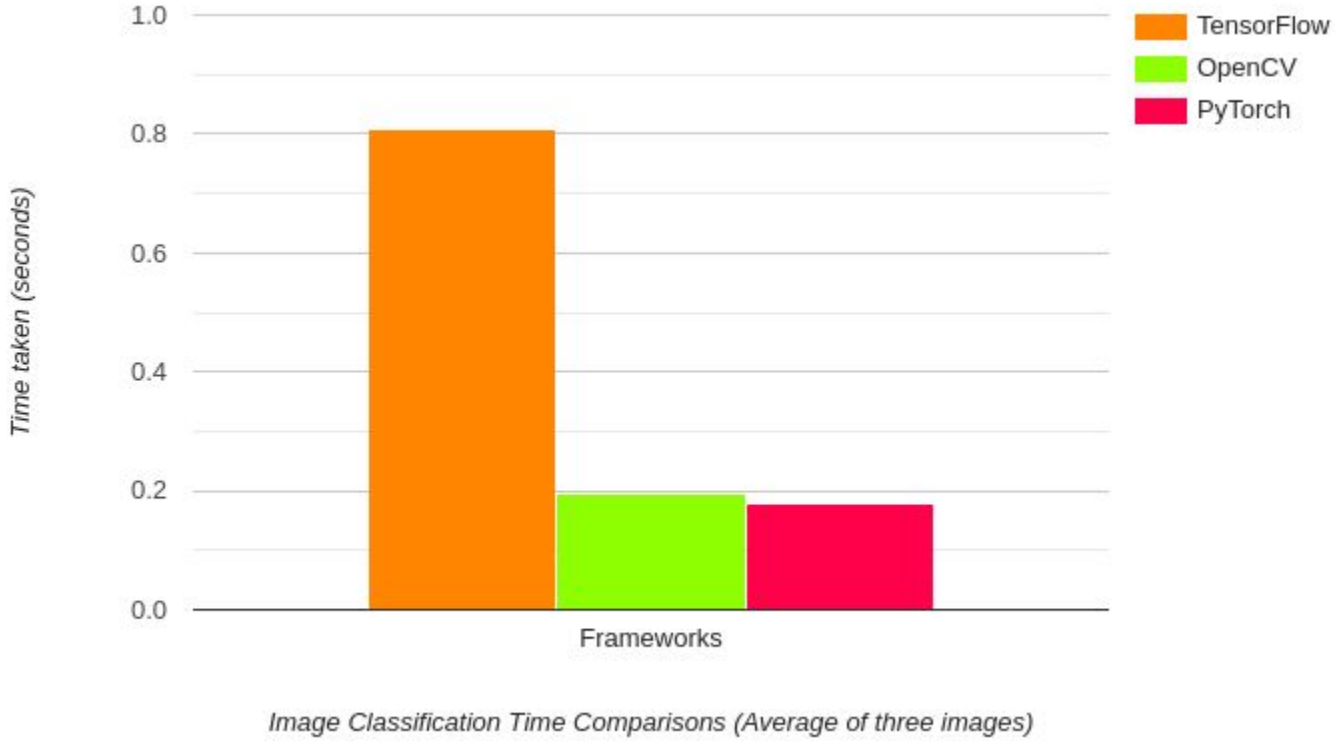
```
$ rosrun my_ros_opencv my_face_detect_opencv_ros.py
```

เปิดหน้าต่างใหม่

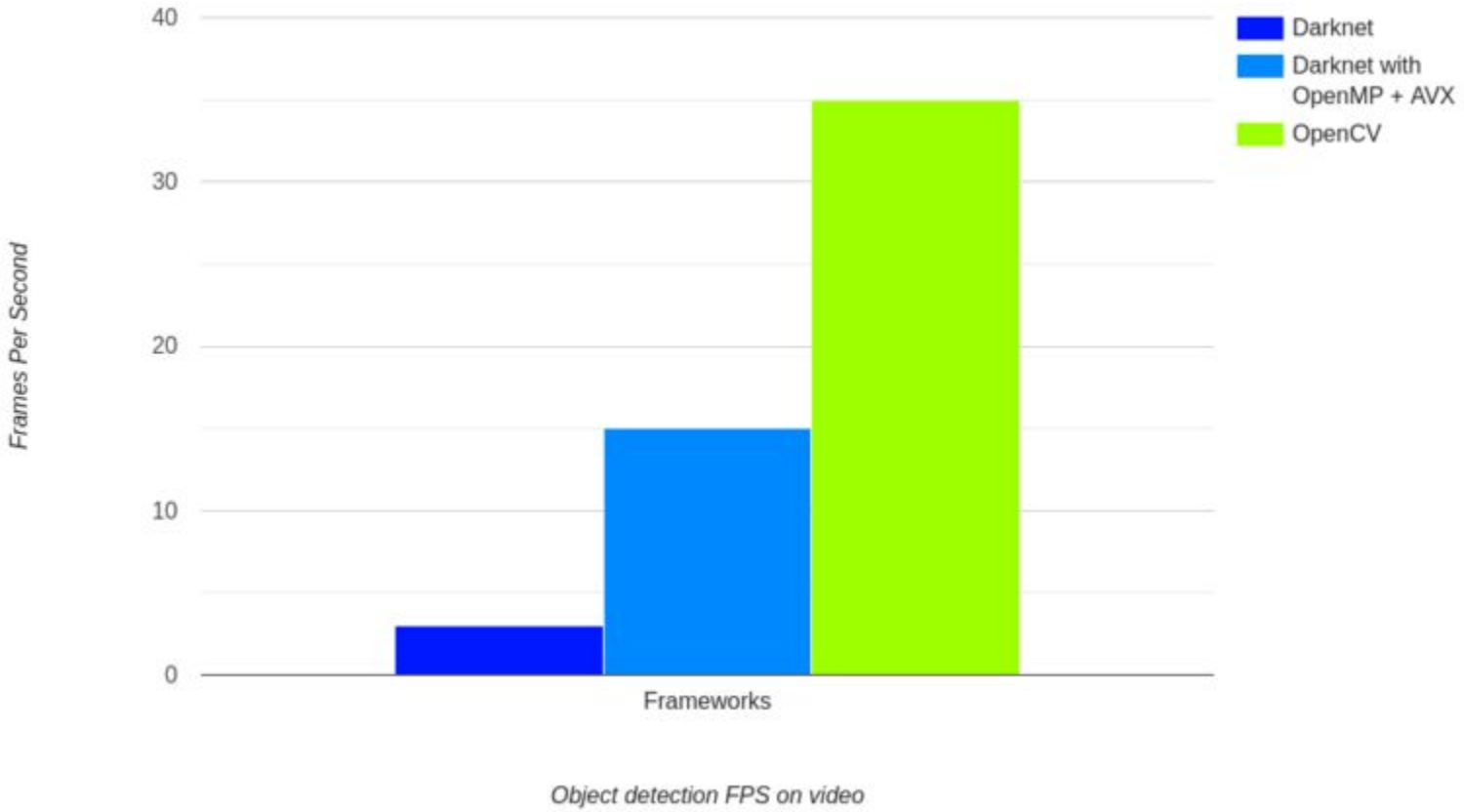
```
$ rosrun my_ros_opencv get_image.py
```


frame@c0665c07a68b

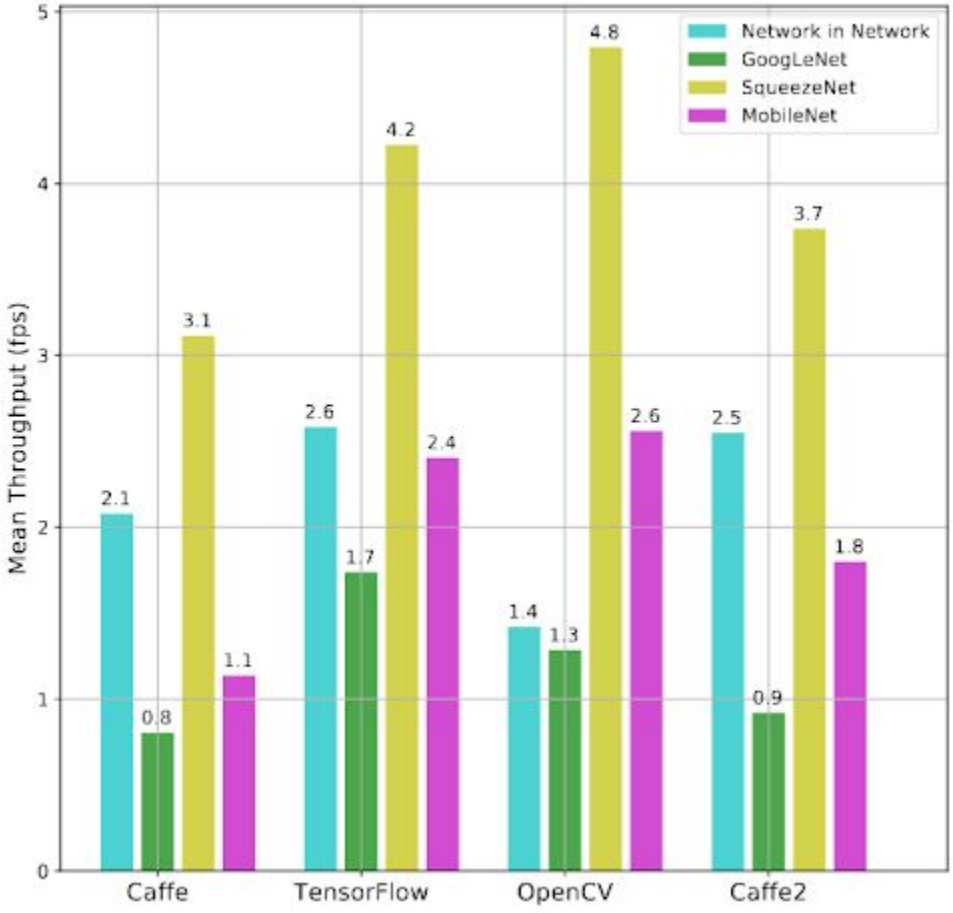
OpenCV DNN



REF: https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide/



REF: https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide/



REF: https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide/

OpenCV DNN

Supported framework

- Caffe
- TensorFlow
- Torch and Pytorch (ONNX model)
- Darknet (YOLO model)

OpenCV DNN with Darknet

```
$ sudo apt install python3-pip
```

```
$ pip3 install imutils
```

```
$ roscd my_ros_opencv
```

```
$ mkdir yolo
```

```
$ cd yolo
```

```
$ wget https://gitlab.com/npttn/file-sharing/-/raw/main/hand-yolov3-tiny.cfg
```

```
$ wget https://gitlab.com/nptttn/file-sharing/-/raw/main/hand-yolov3-tiny_last.weights
```

```
$ cd ../src
```

```
$ gedit opencv_dnn.py
```

opencv_dnn.py

```
#!/usr/bin/env python3

import cv2
import time
import imutils
import argparse
import numpy as np
from imutils.video import FPS
from imutils.video import VideoStream
import rospkg
```

```
12 rospack = rospkg.RosPack()
```

```
PROTOTXT = (rospack.get_path('my_ros_opencv')+'/yolo/hand-yolov3-tiny.cfg') ################################ cha
MODEL = (rospack.get_path('my_ros_opencv')+'/yolo/hand-yolov3-tiny_last.weights') ################################
confidence = 0.5
threshold = 0.3
#Initialize Objects and corresponding colors which the model can detect
labels = ["hand"] ############################## put labels name ####################################
colors = np.random.uniform(0, 255, size=(len(labels), 3))
#Loading Model
print('[Status] Loading Model...')
net = cv2.dnn.readNetFromDarknet(PROTOTXT, MODEL)
# Get the ouput layer names
layer_names = net.getLayerNames()
layer_names = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
cap = cv2.VideoCapture(0)

while True:
    rec, image = cap.read()
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    boxes, confidences, classIDs, idxs = make_prediction(net, layer_names, labels, image, confidence, threshold)
    image = draw_bounding_boxes(image, boxes, confidences, classIDs, idxs, colors)
    cv2.imshow("cap",image)

    if cv2.waitKey(1) == 27:
        break
```

```python
def make_prediction(net, layer_names, labels, image, confidences, threshold):
    height, width = image.shape[:2]

    # Create a blob and pass it through the model
    blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True, crop=False)
    net.setInput(blob)
    outputs = net.forward(layer_names)

    # Extract bounding boxes, confidences and classIDs
    boxes, confidences, classIDs = extract_boxes_confidences_classids(outputs, confidence, width, height)

    # Apply Non-Max Suppression
    idxs = cv2.dnn.NMSBoxes(boxes, confidences, confidence, threshold)

    return boxes, confidences, classIDs, idxs
```

```python
def extract_boxes_confidences_classids(outputs, confidence, width, height):
    boxes = []
    confidences = []
    classIDs = []

    for output in outputs:
        for detection in output:
            # Extract the scores, classid, and the confidence of the prediction
            scores = detection[5:]
            classID = np.argmax(scores)
            conf = scores[classID]

            # Consider only the predictions that are above the confidence threshold
            if conf > confidence:
                # Scale the bounding box back to the size of the image
                box = detection[0:4] * np.array([width, height, width, height])
                centerX, centerY, w, h = box.astype('int')

                # Use the center coordinates, width and height to get the coordinates of the top left corner
                x = int(centerX - (w / 2))
                y = int(centerY - (h / 2))

                boxes.append([x, y, int(w), int(h)])
                confidences.append(float(conf))
                classIDs.append(classID)

    return boxes, confidences, classIDs
```

```python
def make_prediction(net, layer_names, labels, image, confidences, threshold):
    height, width = image.shape[:2]

    # Create a blob and pass it through the model
    blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True, crop=False)
    net.setInput(blob)
    outputs = net.forward(layer_names)

    # Extract bounding boxes, confidences and classIDs
    boxes, confidences, classIDs = extract_boxes_confidences_classids(outputs, confidence, width, height)

    # Apply Non-Max Suppression
    idxs = cv2.dnn.NMSBoxes(boxes, confidences, confidence, threshold)

    return boxes, confidences, classIDs, idxs
```

```
PROTOTXT = (rospack.get_path('my_ros_opencv')+'/yolo/hand-yolov3-tiny.cfg') ################################ cha
MODEL = (rospack.get_path('my_ros_opencv')+'/yolo/hand-yolov3-tiny_last.weights') ################################
confidence = 0.5
threshold = 0.3
#Initialize Objects and corresponding colors which the model can detect
labels = ["hand"] ############################## put labels name ####################################
colors = np.random.uniform(0, 255, size=(len(labels), 3))
#Loading Model
print('[Status] Loading Model...')
net = cv2.dnn.readNetFromDarknet(PROTOTXT, MODEL)
# Get the ouput layer names
layer_names = net.getLayerNames()
layer_names = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
cap = cv2.VideoCapture(0)

while True:
    rec, image = cap.read()
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    boxes, confidences, classIDs, idxs = make_prediction(net, layer_names, labels, image, confidence, threshold)
    image = draw_bounding_boxes(image, boxes, confidences, classIDs, idxs, colors)
    cv2.imshow("cap",image)

    if cv2.waitKey(1) == 27:
        break
```

```python
def draw_bounding_boxes(image, boxes, confidences, classIDs, idxs, colors):
    if len(idxs) > 0:
        for i in idxs.flatten():
            # extract bounding box coordinates
            x, y = boxes[i][0], boxes[i][1]
            w, h = boxes[i][2], boxes[i][3]
            # draw the bounding box and label on the image
            color = [int(c) for c in colors[classIDs[i]]]
            cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)
            text = "{}: {:.4f}".format(labels[classIDs[i]], confidences[i])
            cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    return image
```

```
PROTOTXT = (rospack.get_path('my_ros_opencv')+'/yolo/hand-yolov3-tiny.cfg') ################################ cha
MODEL = (rospack.get_path('my_ros_opencv')+'/yolo/hand-yolov3-tiny_last.weights') ################################
confidence = 0.5
threshold = 0.3
#Initialize Objects and corresponding colors which the model can detect
labels = ["hand"] ############################## put labels name ####################################
colors = np.random.uniform(0, 255, size=(len(labels), 3))
#Loading Model
print('[Status] Loading Model...')
net = cv2.dnn.readNetFromDarknet(PROTOTXT, MODEL)
# Get the ouput layer names
layer_names = net.getLayerNames()
layer_names = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
cap = cv2.VideoCapture(0)

while True:
    rec, image = cap.read()
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    boxes, confidences, classIDs, idxs = make_prediction(net, layer_names, labels, image, confidence, threshold)
    image = draw_bounding_boxes(image, boxes, confidences, classIDs, idxs, colors)
    cv2.imshow("cap",image)

    if cv2.waitKey(1) == 27:
        break
```