Robot Vision II

With Python and OpenCV

Dr. Kourosh Pahlavan

Professor of the Practice, Robotics

L2

Robot Vision II: Embedded Vision

- L1: Course Intro, Jetson Nano, Linux, Package Installations, group building
- L2: Basic Image Processing with Python on Jetson Nano
- L3: Camera Geometry, Motor Control and Interfaces + L2Q
- L4: Project Announcement and Workshop + L3Q

Your robot must be fully operational at this point

- L5: Project Workshop + GHWA1
- L6: Project Workshop + GHWA2
- L7: Project Workshop + GHWA3
- L8: Project Workshop + GHWA4
- L9: Hardware Disclosures and Formal Design Reviews
- L10: Robot Soccer Tournament

Upgrade OpenCV to OpenCV_contrib

- Download the following shell script:

  https://github.com/AastaNV/JEP/blob/master/script/install_opencv4.5.0_Jetson.sh

- Make sure the file is executable:

```
$ chmod u+x install_opencv4.5.0_Jetson.sh
```

- Some 12 hours later, the installation is hopefully over:

```
$ …
Python3
>>> import cv2
>>> print(cv2.getBuildInformation())
```

- Try any of your OpenCV programs from Robot Vision I
  - For cases with live camera input see the following slides

Jetson Nano: Camera Module

- USB webcam
  - USB interface
- CSI-MIPI camera (IMX219 Raspberry Pi V2). Direct interface -> faster

```
$ sudo apt-get update
$ sudo apt-get install v4l-utils
$ v4l2-ctl --list-formats-ext
```

```
ioctl: VIDIOC_ENUM_FMT
            Index       : 0
            Type        : Video Capture
            Pixel Format: 'RG10'
            Name        : 10-bit Bayer RGRG/GBGB
                        Size: Discrete 3280x2464 Interval: Discrete 0.048s (21.000 fps)
                        Size: Discrete 3280x1848 Interval: Discrete 0.036s (28.000 fps)
                        Size: Discrete 1920x1080 Interval: Discrete 0.033s (30.000 fps)
                        Size: Discrete 1280x720 Interval: Discrete 0.017s (60.000 fps)
                        Size: Discrete 1280x720 Interval: Discrete 0.017s (60.000 fps)
```

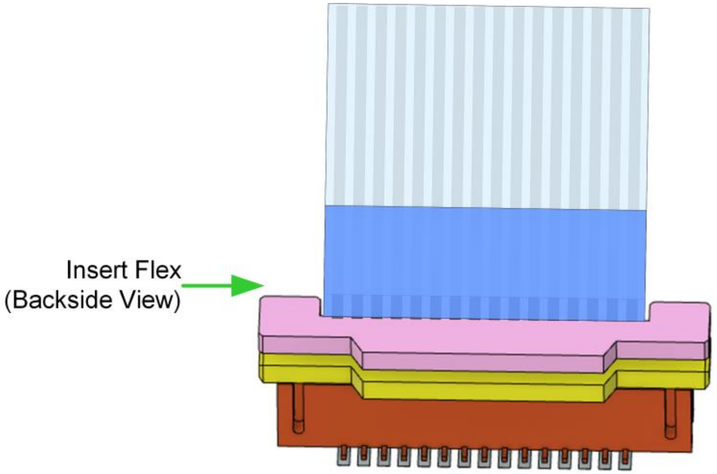
Camera flip parameters

```
flip-method: video flip methods
             flags: readable, writable, controllable
             Enum "GstNvVideoFlipMethod" Default: 0, "none"
                 (0): none                - Identity (no rotation)
                 (1): counterclockwise - Rotate counter-clockwise 90 degrees
                 (2): rotate-180          - Rotate 180 degrees
                 (3): clockwise           - Rotate clockwise 90 degrees
                 (4): horizontal-flip   - Flip horizontally
                 (5): upper-right-diagonal - Flip across upper right/lower left diagonal
                 (6): vertical-flip     - Flip vertically
                 (7): upper-left-diagonal - Flip across upper left/low
```

Camera Installation (Jetson Nano)

- Types of interfaces supported by Jetpack:
  - MIPI CSI-2 interface (two camera interfaces):
    - Raspberry Pi Camera Module V2: IMX219
  - USB interface:
    - Webcams
  - Ethernet
    - IP-cams
- Install IMX219:
  - Gently lift the top bracket (1-2 mm)
  - Insert the ribbon the contact fingers facing the board (the blue tape faces outwards)
- More info at:

  https://developer.nvidia.com/embedded/learn/jetson-nano-2gb-devkit-user-guide#id-.JetsonNano2GBDeveloperKitUserGuidevbatuu_v1.0-Camera



Camera Access (Jetson Nano)

- "nvgstcapture" is the app using "gstreamer" to display images/videos:

```
$ nvgstcapture-1.0 --help
# CSI camera:
$ nvgstcapture-1.0 --orientation 2

# V4L2 USB camera (see /dev/video<#>):
$ nvgstcapture-1.0 --camsrc=0 --cap-dev-node=<#>
```

- Camera operation:
  - Press 'j' to Capture one image.
  - Press 'q' to exit
- Capture video and save to disk:

```
$ nvgstcapture-1.0 --mode=2
```

  - Press '1' to Start recording video
  - Press '0' to Stop recording video
  - Press 'q' to exit
- More info at:

https://developer.nvidia.com/embedded/learn/tutorials/first-picture-csi-usb-camera

Access CSI Camera in Python

```
def gstreamer_pipeline(sensor_id=0,
    capture_width=1920,
    capture_height=1080,
    display_width=960,
    display_height=540,
    framerate=30,
    flip_method=0,):
    return ("nvarguscamerasrc sensor-id=%d !"
        "video/x-raw(memory:NVMM), width=(int)%d, height=(int)%d, framerate=(fraction)%d/1 ! "
        "nvvidconv flip-method=%d ! "
        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
        "videoconvert ! "
        "video/x-raw, format=(string)BGR ! appsink"
        % (sensor_id,
            capture_width,
            capture_height,
            framerate,
            flip_method,
            display_width,
            display_height,)
    )
```

Object Detection with CUDA Directly

```
import jetson.inference
import jetson.utils

net = jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.5)
camera = jetson.utils.videoSource("csi://0")       # '/dev/video0' for V4L2
display = jetson.utils.videoOutput("display://0") # 'my_video.mp4' for file

while display.IsStreaming():
    img = camera.Capture() # Note: img is in CUDA format
    detections = net.Detect(img)
    display.Render(img)
    display.SetStatus("Object Detection | Network\
    {:.0f} FPS".format(net.GetNetworkFPS()))import jetson.inference
```

Object Detection with OpenCV and CUDA…

```
import jetson.inference
import jetson.utils
import cv2
from gsp import gstreamer_pipeline as gsp
net = jetson.inference.detectNet('ssd-mobilenet-v2', threshold = 0.5)
def main():
    window_title = "CSI Camera"
    cap = cv2.VideoCapture(gsp(flip_method=0), cv2.CAP_GSTREAMER)
    if cap.isOpened():
        try:
            window_handle = cv2.namedWindow(window_title, cv2.WINDOW_AUTOSIZE)
            while True:
                ret, img = cap.read()
                imgCuda = jetson.utils.cudaFromNumpy(img)
                # the reverse is done by: img = jetson.utils.cudaToNumpy(imgCuda)
                detections = net.Detect(imgCuda)
```

…Object Detection with OpenCV and CUDA

```
                for d in detections:
                    xt, yt, xb, yb = int(d.Left), int(d.Top), int(d.Right), int(d.Bottom)
                    className = net.GetClassDesc(d.ClassID)
                    cv2.rectangle(img, (xt, yt), (xb, yb), (255, 0, 0), 2)
                    cv2.putText(img, className, (xt+5, yt+1), cv2.FONT_HERSHEY_DUPLEX, 0.75,\
                    (255, 255, 0), 2)
                cv2.imshow(window_title, img)
                keyCode = cv2.waitKey(10) & 0xFF
                # Stop the program on the ESC key or 'q'
                if keyCode == 27 or keyCode == ord('q'):
                    break
            # end of while…
        finally:# try…
            cap.release()
            cv2.destroyAllWindows()
    else: # if cap…
        print("Error: Unable to open CSI camera")

if __name__ == "__main__":
    main()
```

Custom Dataset and Model Retraining

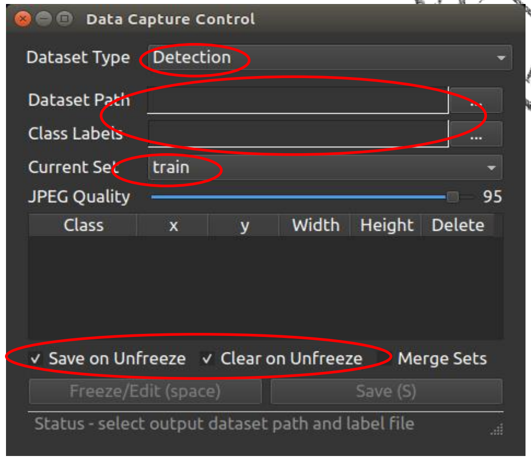
- Create file "labels.txt" under jetson-inference/python/training/detection/ssd/data

```
Baller-bot
Goal
…
```

- Run camera-capture to collect images from different directions and backgrounds:

```
$ camera-capture csi://0        # using default MIPI CSI camera
$ camera-capture /dev/video0    # using V4L2 camera /dev/video0
```

  - Press 'Freeze', draw a tight frame around th eobject and press 'Save'. Continue…



Data Capture Control

Dataset Type: Detection

Dataset Path: ...

Class Labels: ...

Current Set: train

JPEG Quality: 95

Class	x	y	Width	Height	Delete

✓ Save on Unfreeze ✓ Clear on Unfreeze Merge Sets

Status - select output dataset path and label file

- Train your model:Automated operation:

```
$ cd jetson-inference/python/training/detection/ssd
$ python3 train_ssd.py --dataset-type=voc --data=data/<YOUR-DATASET> --model-dir=models/<YOUR-MODEL>
```

- Convert the trained PyTorch model to ONNX and load it with detectnet:

```
$ python3 onnx_export.py --model-dir=models/<YOUR-MODEL>
NET=models/<YOUR-MODEL>
detectnet --model=$NET/ssd-mobilenet.onnx --labels=$NET/labels.txt --input-blob=\
input_0 --output-cvg=scores --output-bbox=boxes csi://0
```

- More info at:
  https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-collect-detection.md