V4L2 & VPI

Video Reading and Capture

【110上】嵌入式系統技術實驗課程

TA: 陳翰群 hanz1211.ee09@nycu.edu.tw

V4L2 API

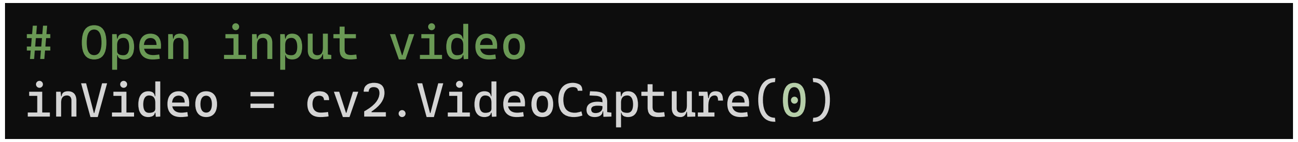
- V4L2 is a video capture API named “video for Linux version 2”
- A kernel interface for analog radio and video capture and output drivers
- Webcam, TV Card
- V4L2 API Specification
  https://www.kernel.org/doc/html/v4.9/media/uapi/v4l/v4l2.html

V4L2 Capture Flow

1. Open video device
2. Ask video capability
3. Set video device capture format
4. Mapping framebuffer to address
5. Start capturing
6. Deque(double side queue) a framebuffer to get image

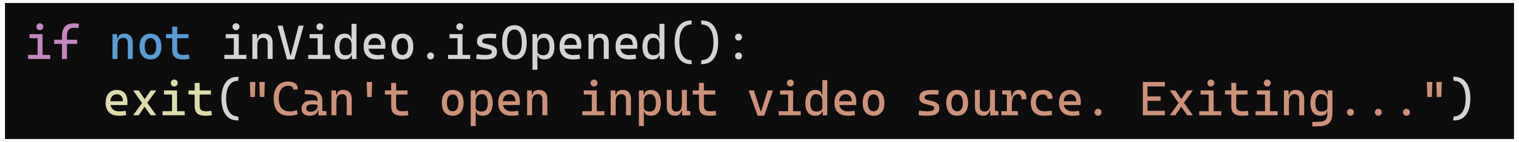
Video Capture in OpenCV

1. Declare a video source (/dev/video0)



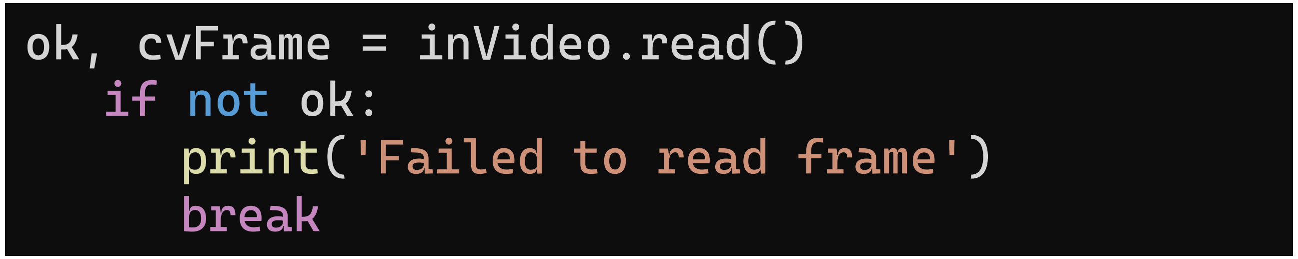
```
# Open input video
inVideo = cv2.VideoCapture(0)
```

2. Check if the video source is opened:



```
if not inVideo.isOpened():
   exit("Can't open input video source. Exiting...")
```

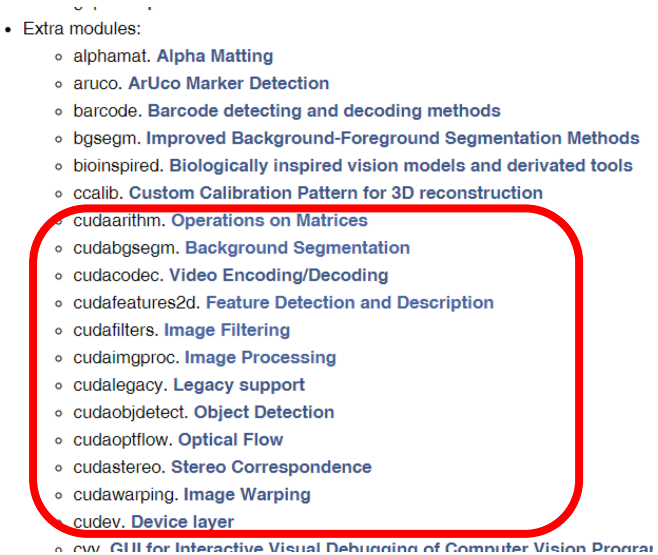
3. Read one frame to cvFrame:



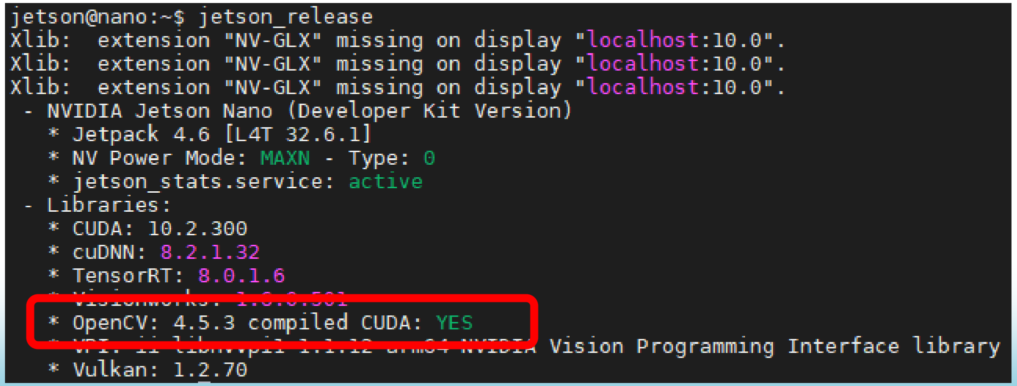
```
ok, cvFrame = inVideo.read()
   if not ok:
      print('Failed to read frame')
      break
```

OpenCV CUDA Support

- Default OpenCV installation does not activate CUDA GPU Support
  - The Jetson Nano Image provided has compiled OpenCV with CUDA for you
- In official documentation, all CUDA util are listed in extra modules, includes:
  - Matrix operation (arithmetic, convolution)
  - Video Encode/Decode
  - Feature (FAST, ORB)
  - Image Processing
  - Object Detection
  - Optical Flow
- Not all cv2 functions are supported
- You can use `jetson_release` to check if OpenCV compiled with CUDA



- Extra modules:
  - alphamat. Alpha Matting
  - aruco. ArUco Marker Detection
  - barcode. Barcode detecting and decoding methods
  - bgsegm. Improved Background-Foreground Segmentation Methods
  - bioinspired. Biologically inspired vision models and derivated tools
  - ccalib. Custom Calibration Pattern for 3D reconstruction
  - cudaarithm. Operations on Matrices
  - cudabgsegm. Background Segmentation
  - cudacodec. Video Encoding/Decoding
  - cudafeatures2d. Feature Detection and Description
  - cudafilters. Image Filtering
  - cudaimgproc. Image Processing
  - cudalegacy. Legacy support
  - cudaobjdetect. Object Detection
  - cudaoptflow. Optical Flow
  - cudastereo. Stereo Correspondence
  - cudawarping. Image Warping
  - cudev. Device layer
  - cvv. GUI for Interactive Visual Debugging of Computer Vision Progra

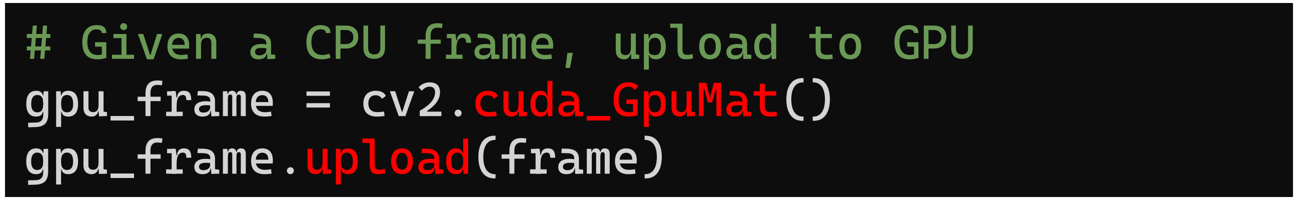


```
jetson@nano:~$ jetson_release
Xlib:  extension "NV-GLX" missing on display "localhost:10.0".
Xlib:  extension "NV-GLX" missing on display "localhost:10.0".
Xlib:  extension "NV-GLX" missing on display "localhost:10.0".
 - NVIDIA Jetson Nano (Developer Kit Version)
   * Jetpack 4.6 [L4T 32.6.1]
   * NV Power Mode: MAXN - Type: 0
   * jetson_stats.service: active
 - Libraries:
   * CUDA: 10.2.300
   * cuDNN: 8.2.1.32
   * TensorRT: 8.0.1.6
   * Visionworks: 1.6.0.501
   * OpenCV: 4.5.3 compiled CUDA: YES
   * VPI: ii libnvvpi1 1.1.12 arm64 NVIDIA Vision Programming Interface library
   * Vulkan: 1.2.70
```

NCTU iVSLAB CONFIDENTIAL DOCUMENT DO NOT COPY OR DISTRIBUTE

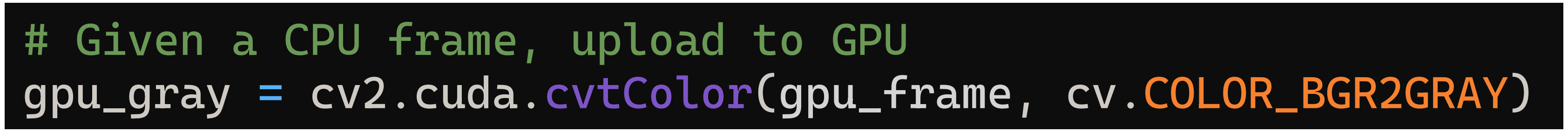
OpenCV CUDA Support

- GPU ram (VRAM) and CPU memory (DRAM) are not shared in system
  - Need to create a dedicated matrix for GPU operation



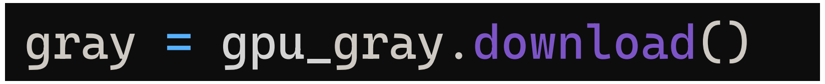
```
# Given a CPU frame, upload to GPU
gpu_frame = cv2.cuda_GpuMat()
gpu_frame.upload(frame)
```

  - You can operate similar functions if they are supported



```
# Given a CPU frame, upload to GPU
gpu_gray = cv2.cuda.cvtColor(gpu_frame, cv.COLOR_BGR2GRAY)
```

  - To show the result or other I/O related functions, you need to pass matrix back to CPU



```
gray = gpu_gray.download()
```

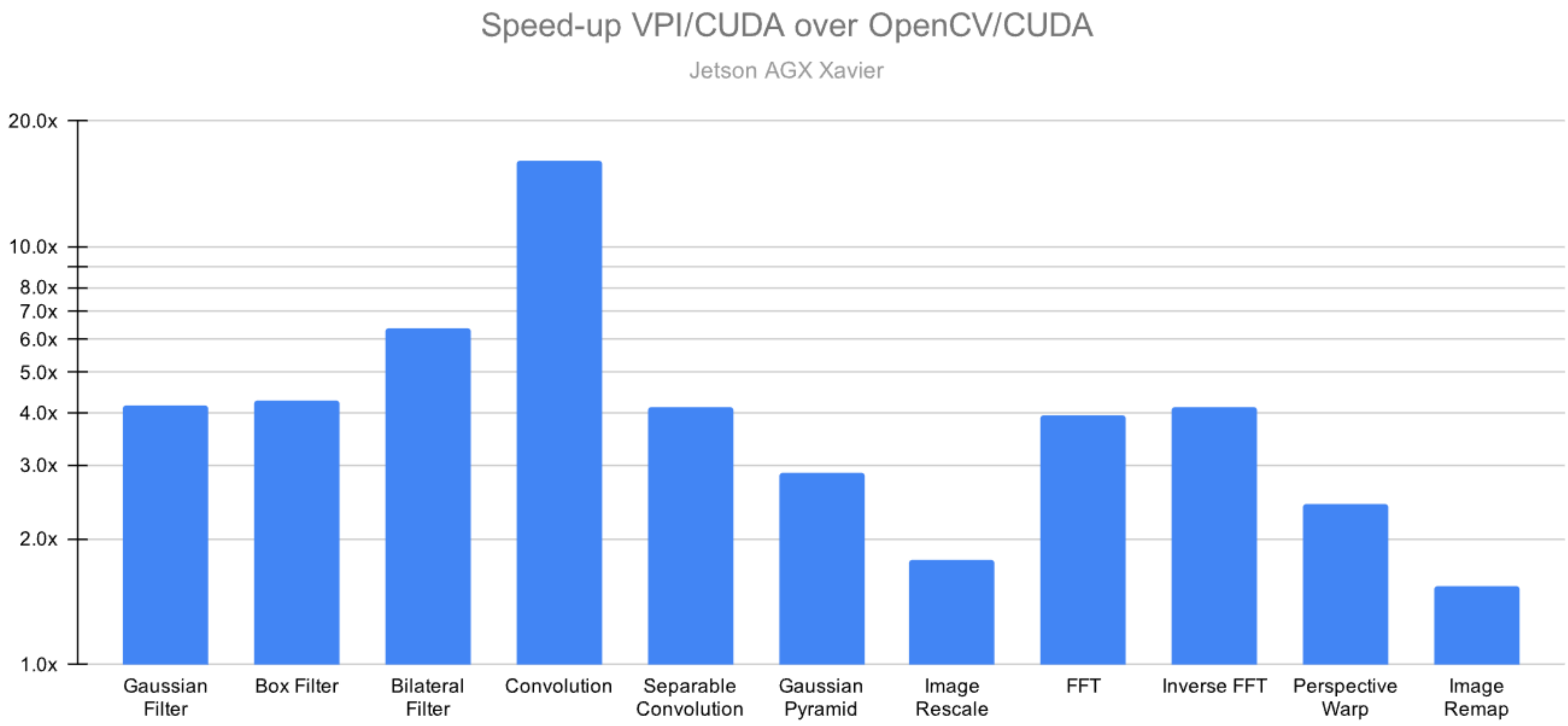
NCTU iVSLAB CONFIDENTIAL DOCUMENT DO NOT COPY OR DISTRIBUTE

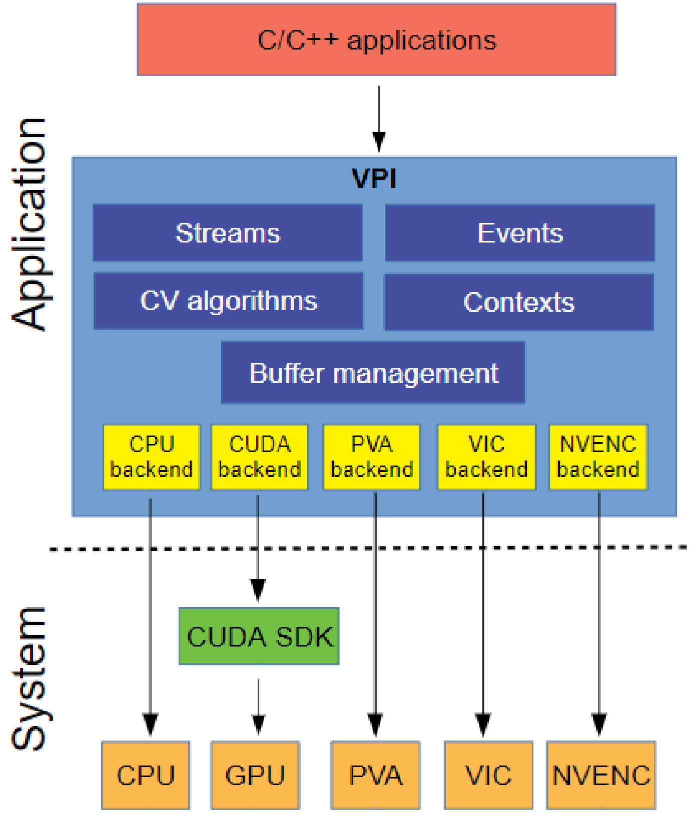
OpenCV CUDA Support

- Some detailed examples using OpenCV w/ CUDA
  - https://medium.com/dropout-analytics/opencv-cuda-for-videos-f3dcf346e398
  - https://learnopencv.com/getting-started-opencv-cuda-module/

Vision Programming Interface

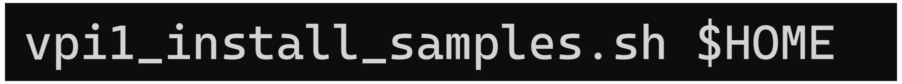
- VPI - Vision Programming Interface
- A new library with built-in functions to utilize GPU with NVIDIA CUDA sdk
- A basic performance comparison with OpenCV on GPU:





Vision Programming Interface

- Official Documentation: https://docs.nvidia.com/vpi/index.html
- There are some sample applications you can try:
  - Installation: https://docs.nvidia.com/vpi/installation.html
  - After that, copy the samples to home directory, run this in terminal:



```
vpi1_install_samples.sh $HOME
```

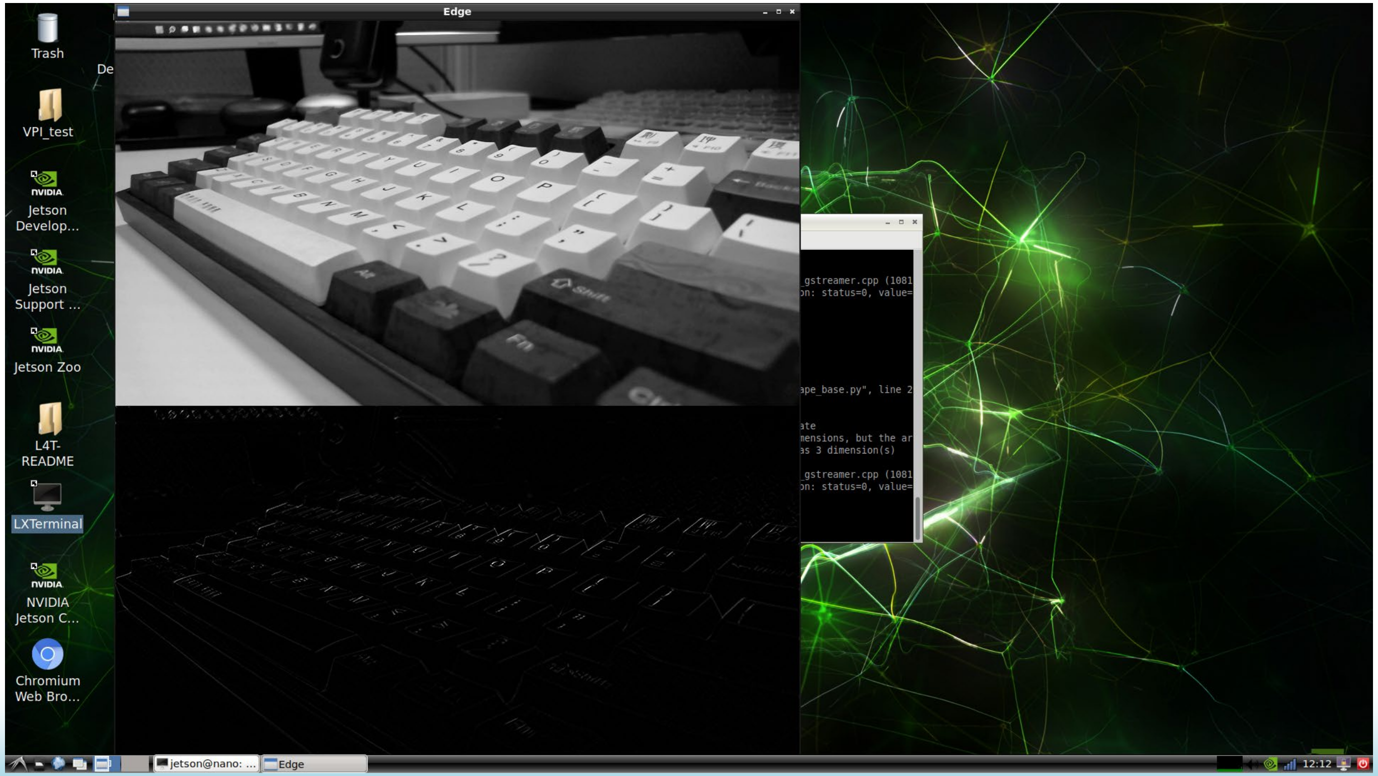
  - You should see NVIDIA_VPI–samples in you home directory (/home/jetson)
  - There are 14 samples app but not all of them supported
    - Check the supported algorithm: https://docs.nvidia.com/vpi/algorithms.html
    - On Jetson Nano, only CPU, CUDA and some VIC backend are supported

Example: VPI Edge Detection

- edge_vpi.py
- Run the edge detection in CPU or CUDA (connect webcam first)

```
python3 edge_vpi.py cpu
```

```
python3 edge_vpi.py cuda
```



```
_gstreamer.cpp (1081
on: status=0, value=

ape_base.py", line 2

ate
mensions, but the ar
as 3 dimension(s)

_gstreamer.cpp (1081
on: status=0, value=
```

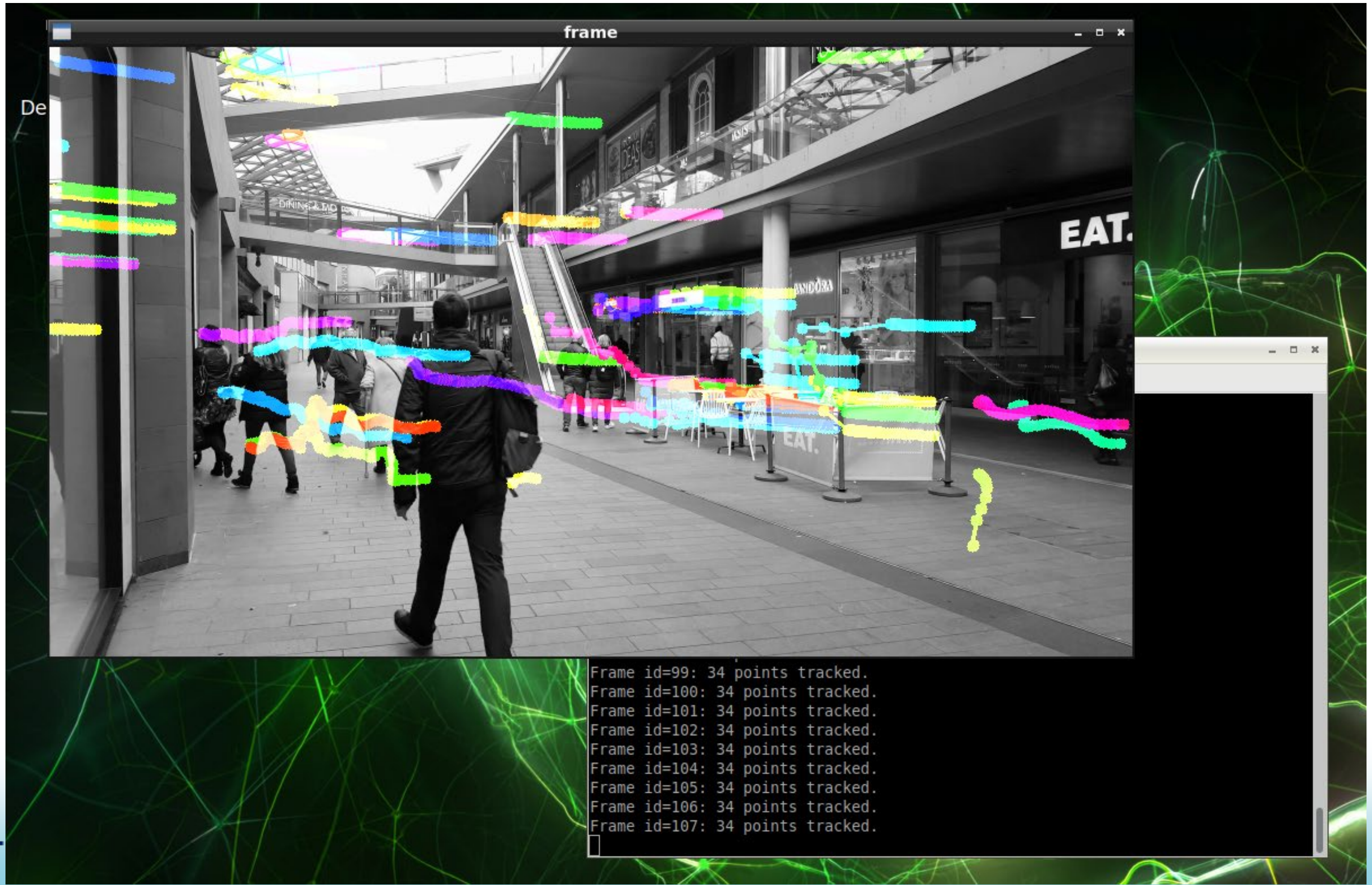
NCTU iVSLAB CONFIDENTIAL DOCUMENT DO NOT COPY OR DISTRIBUTE

Example: Pyramidal LK Optical Flow

- pyramidal.py

```
python3 pyramidal.py
```

- Modified from official sample application, track feature points and draw with OpenCV



```
Frame id=99: 34 points tracked.
Frame id=100: 34 points tracked.
Frame id=101: 34 points tracked.
Frame id=102: 34 points tracked.
Frame id=103: 34 points tracked.
Frame id=104: 34 points tracked.
Frame id=105: 34 points tracked.
Frame id=106: 34 points tracked.
Frame id=107: 34 points tracked.
```