PyQt for CV

GUI Applications for Computer Vision

【110上】嵌入式系統技術實驗課程

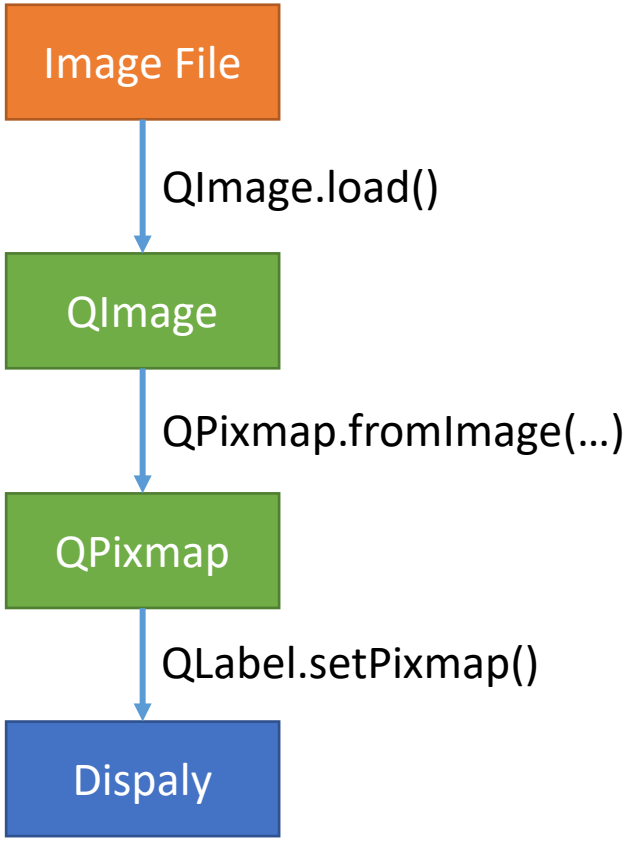
TA: 陳翰群 hanz1211.ee09@nycu.edu.tw

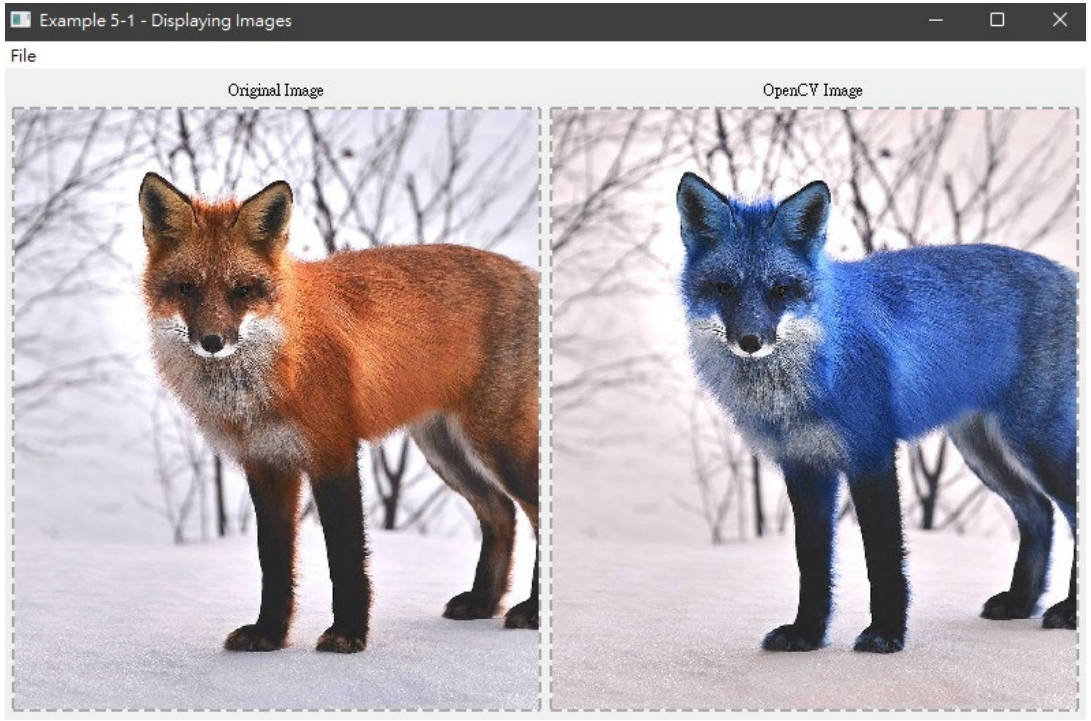
Display Image from OpenCV in PyQt

- PyQt has QPixmap and QImage for managing image data.
  - If you only need to read or write an image without manipulating the file, then create a QPixmap object and call QLabel's setPixmap()
  - If you need to modify an image's data, you will need to convert from QPixmap to QImage, perform the operations, and then convert back to QPixmap to show the image.

Display Image from OpenCV in PyQt

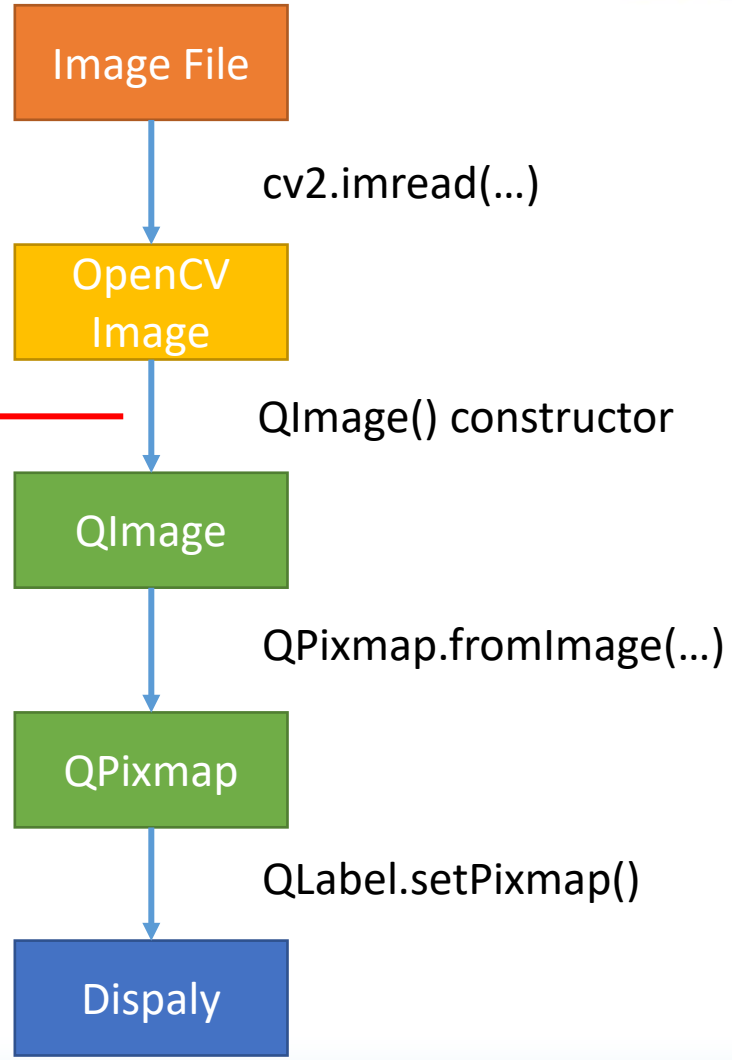
- display_images.py





Without conversion

Need to convert from BGR to RGB before declaring QImage



Display Image from OpenCV in PyQt

- display_images.py
- Two ways to convert colors:
  1. When creating a QImage object, pass QImage.Format_BGR888 as an argument to reverse the colors from BGR to RGB.

```
QImage(cv_image, width, height, bytes_per_line, QImage.Format_RGB888)
```

```
QImage(cv_image, width, height, bytes_per_line, QImage.Format_BGR888)
```

  2. After an image is loaded using imread(), use cv2.cvtColor(image, cv2.COLOR_BGR2RGB) on the Mat.

- Choose only one and be consistent throughout the project

Display Image from OpenCV in PyQt

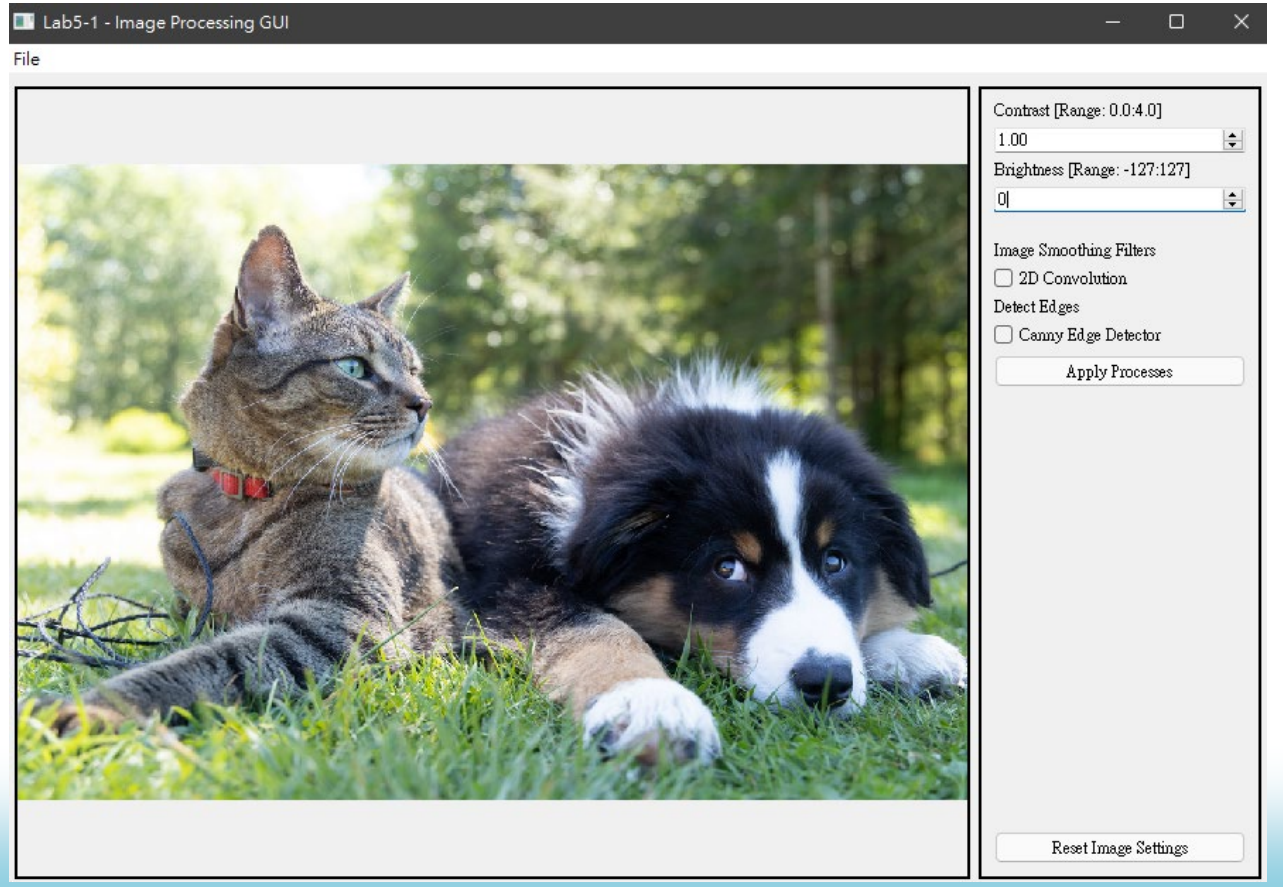
- Another conversion method is written in lab5-1.py
  - Note that this function directly change QPixmap content
    - You may change it to a static function for reuse, not needed to complete this lab.

```python
def convertCVToQImage(self, image):
    """Load a cv image and convert the image to a Qt QImage. Display the image in image_label."""
    cv_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Get the shape of the image, height * width * channels. BGR/RGB/HSV images have 3 channels
    height, width, channels = cv_image.shape  # Format: (rows, columns, channels)
    # Number of bytes required by the image pixels in a row; dependency on the number of channels
    bytes_per_line = width * channels
    # Create instance of QImage using data from cv_image
    converted_Qt_image = QImage(cv_image, width, height, bytes_per_line, QImage.Format_RGB888)

    self.image_label.setPixmap(QPixmap.fromImage(converted_Qt_image).scaled(
        self.image_label.width(), self.image_label.height(), Qt.KeepAspectRatio))
```

Here is the resize done with QPixmap.scaled, an alternative if you did not use cv2.resize()

Lab5-1 - Image Processing GUI

- lab5-1.py
  - Follow the TODOs and complete this program
  - This is a quick recap for both PyQt and OpenCV
- You should be able to:
  - Open any image
  - Select and apply changes
  - Reset all changes
  - Save the modified image



Display Video and Multithreading

- When a PyQt application is started using exec_(), the program's event loop begins.
  - Starting the event loop also creates a thread, known as the main thread
  - All events that occur in the application, such as clicking a button or typing text in line edit widgets, will be handled sequentially using your computer's CPU and other resources.
- Opening and displaying videos from OpenCV will cause our application to become hung up as more resources are needed.
  - Since your GUI must run in the main thread, we need to create a secondary thread, also known as a worker thread
  - This will unload some of the extra processing work from the main thread and keep our application responsive

QThread and Custom Signal

- display_video.py
- Overwrite QThread.run() as worker thread main sequence
  - Use a while loop to read video frame continuously without blocking main thread
- For communication between main/worker thread, use signal & slot
  - Declare custom signal in worker, need to import from PyQt5.QtCore

```
frame_data_updated = pyqtSignal(ndarray)
```

  - Connect worker's signal to main's slot, connection done in main thread

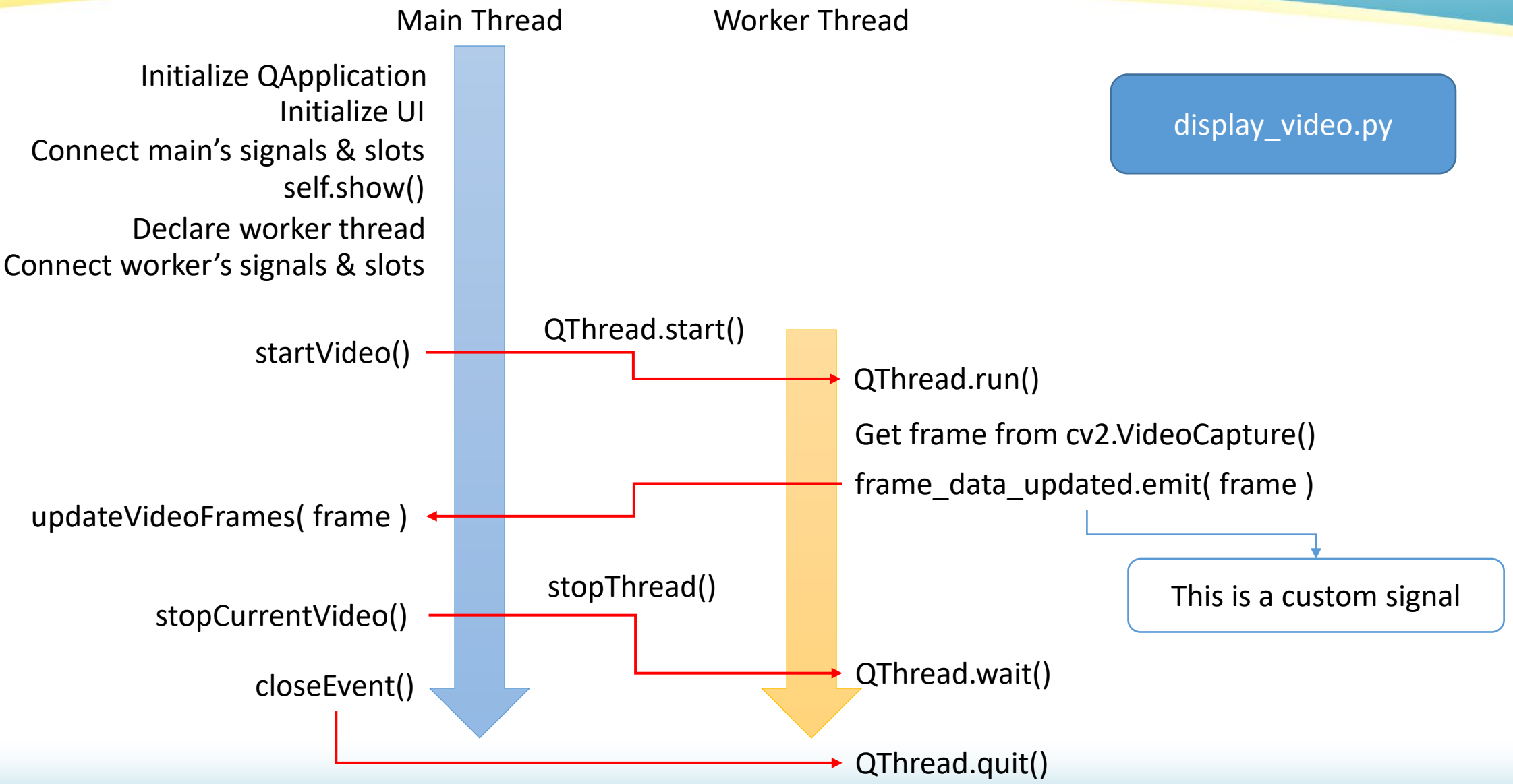
```
self.video_thread_worker.frame_data_updated.connect(self.updateVideoFrames)
```

  - Activate signal with emit(), you can pass the corresponding type of data (numpy.ndarray)

```
self.frame_data_updated.emit(frame)
```

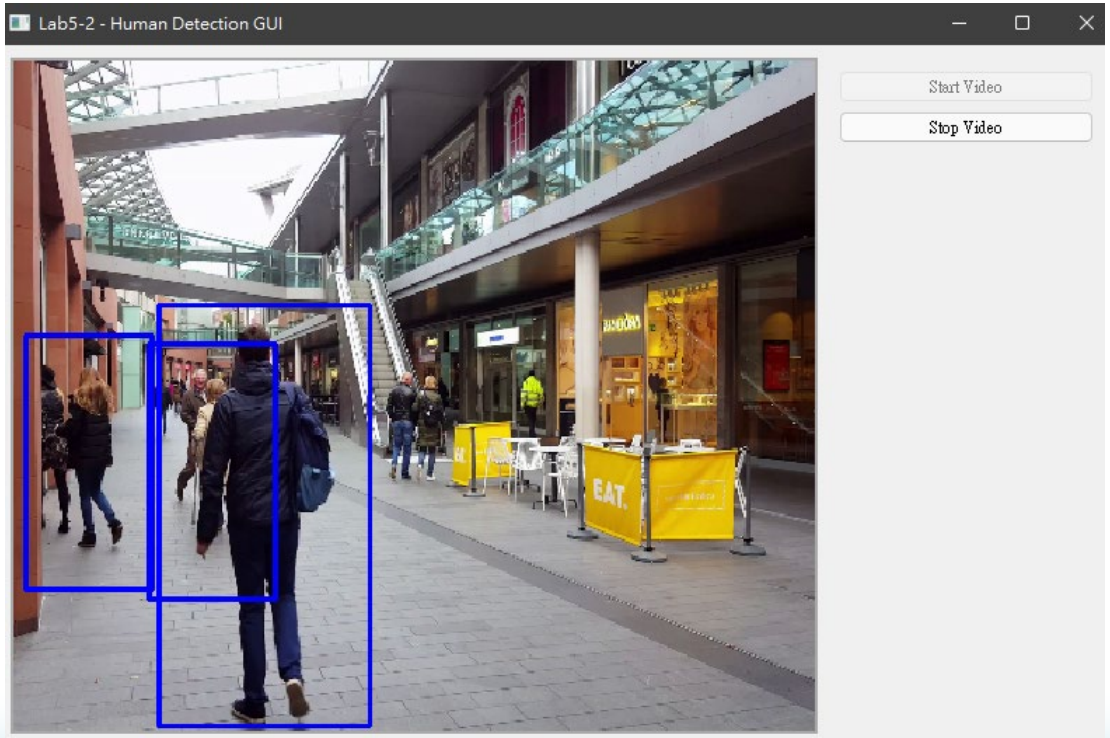
  - In main thread, updateVideoFrames will receive data, and show on video_display_label

Display Video with Multithreading



Lab5-2 Human Detection GUI

- lab5-2.py
- In this lab we will use HOG descriptor for human detection in video
  - You can check createHOGDescriptor if you are interested



Demo

- 本次Lab以個人為單位
- 配分
  - Lab5-1 : 50%
  - Lab5-2 : 50%
- Demo
  - 完成Lab後，請進視訊會議舉手呼叫助教們demo
  - 多個小題可以分次demo
  - 根據助教要求呈現程式執行結果
- 最後登記時間：21:20