EEL 4745C: Microprocessor Applications 2
Lab **5**: AI with RTOS
Fall 2022

OBJECTIVES
In this lab, we will interface with a **Beagle** and set up a simple AI pipeline on it. In particular, We will communicate to a light Python-OpenCV pipeline for real-time face detection: on image data, webcam feed, and video files.

REQUIRED MATERIALS

Hardware
- Beaglebone Black and SD card
- Laptop workstation with Python and OpenCV

Software
- Python and OpenCV
- Skeleton code provided for Lab 5

Documentations
- https://beagleboard.org/getting-started
- https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html
- https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
- https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81
- Lecture-8 contents

IN-LAB REQUIREMENTS
Lab-5 officially starts on the week of (10/31-11/04). Since we pushed back the previous lab by one week, all lab-5 demos are also pushed back (without late penalty); the demos are due on 11/14-11/18.

PRE-LAB QUESTIONS
- Your Beagle has on-board memory; but we are using a SD card as well. Why/when do we use the on-device memory over SD card?
- Install Python and OpenCV libraries on your laptop or workstation. Practice:
  - How to read/write an image.
  - How to draw shapes (circles, rectangles, texts) on an image.

Part A: Interface & communicate with a Beagle

First, we will learn how to set up the software image on the Beagle. We will need a host laptop/computer to connect to the Beagle, then SSH into it, and install the OS components. Please follow the steps outlined at https://beagleboard.org/getting-started. We have listed the simplified steps for both Windows and Mac host machines in **Lecture 8**.

Specifically, you will follow the following steps:
1. Download Putty
2. Download Beagle software image to SD card
3. Download USB Driver
4. Power On and Boot the Beagle
5. SSH Into the Beagle

For MacOS hosts, you will need to do additional steps:
6. Update USB CDC Ethernet to use NCM
7. Connect your Beagle with Mac PC

After that, you will be able to connect Beagle to the internet and set up any AI/AIoT pipelines that you need for your work!

If you have used Beagles or Raspberry Pis before, you probably know these basic steps already. If not, please follow the steps outlined in **Lecture 8** and consult with the TAs if you get stuck with any bugs!

Part B: Setup Python-OpenCV pipeline

Now install Python and OpenCV libraries on Beagle.

>> Get Python (3.8 or 3.9):
```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install python3
```

>> Verify the installation:
```
$ python3 --version
```

>> Get OpenCV 3.2.x:
```
$ sudo apt install python3-opencv
```

>> Verify the installation:
```
$ python3 -c "import cv2;
              print(cv2.__version__)"
```

Part C: On-Board Face detection

We will try to run the standard face detection module of OpenCV on the Beagle. Learning the OpenCV library is not the purpose of this class, and it takes some time. Nevertheless, we will learn the basics of image processing and simple classifiers and object detection pipelines.

In particular, we will learn the basics of face detection using the Haar Feature-based Cascade Classifiers. See the following tutorials:
- https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
- https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81

The steps are the following:

- Read an image file given the path: `image = cv2.imread(img_path)`
  - `image` is now a numpy array of size WxHx1 for grayscale images or WxHx3 for RGB images
  - *Note: OpenCV reads images in BGR format.*
- Load the cascade classifier model: `faceCascade = cv2.CascadeClassifier(cascade_path)`
- Detect faces: `faces = face_detect(image, faceCascade)`
- Detect bounding boxes on the image: `image = draw_face_boxes(image, faces)`
- Write the output image as a file: `cv2.imwrite("output.jpg", image)`
- The **`face_detect`** and **`draw_face_boxes`** functions are given below.

Your tasks:

- Get familiar with OpenCV functionalities on your laptop, run the `face_detect_img.py` and match the output. Try different images and see when the output breaks.
- Also fun the `face_detect_webcam.py` and see if you can detect your face in real time!
- Then copy over the provided code and data to your Beagle.
- The `face_detect_webcam.py` should run fine and generate the right output.
- Your task is to complete the `face_detect_video.py` file so that
  - It takes input a path for a video file
  - Reads the video file frame by frame
  - Detects faces in each frame, and draws bounding boxes if any faces are found
  - The final output should be a video with all the face boxes
  - Check: https://youtu.be/prSROEIjuaY

- Bonus points (1-2):
  - Can you implement some tracking features? (By Kalman filtering or any other filtering approach)

```
def face_detect(image, faceCascade):
    if len(image.shape)==3:
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        gray = image

    # Detect faces in the image
    faces = faceCascade.detectMultiScale(
                        gray,
                        scaleFactor=1.1,
                        minNeighbors=5,
                        minSize=(30, 30),
                        flags=cv2.CASCADE_SCALE_IMAGE
            )
    return faces


def draw_face_boxes(image, faces):
    if faces is None:
        return image
    for (x, y, w, h) in faces:
        cv2.rectangle(image,
                      (x, y), (x+w, y+h), (0, 255, 0), 2
        )
    return image
```