ALBIR OPENMV_BOT MANUAL

Contents

Module repository

https://github.com/dragonflyneuro/ICL_BioEng_ALBIR_OpenMV

Software development notes

General notes

- OpenMV development is done in micropython, which has some important differences to standard python. Some common modules used with python are not available for use.
- Please read through all the classes and functions we provide in our default codebase and have a look through the OpenMV documentation (https://docs.openmv.io/index.html).
- For the first assignment, you will not be able to use your own bots thus you should rely on the default classes and functions we provide.
- For the rest of the course, we highly recommend modifying and experimenting with the default codebase and making it yours – get the most out of your bot!
- It may be helpful to write code to have one of the LEDs on the OpenMV turn on when troubleshooting.
- Do most of your testing over USB cable for ease of development and to keep the Lipo battery charged – your laptop will not be able to access the internet if connecting to the bot wirelessly (see page 4).

Moving files to your robot

When you plug the OpenMV_bot into your computer via USB, you should be able to access the contents of the SD card in the OpenMV as a USB device. All files on board the SD card can safely be deleted if you want a clean slate. **We HIGHLY suggest you edit and run files directly in the SD card to avoid potential problems with cached modules not updating every time you run a script.**

Using OpenMV IDE

All compiling of code is done on-board the OpenMV, so you will not be able to test code when not connected to the OpenMV_bot. OpenMV IDE allows you to connect to the bot over USB cable with ease. The USB button on the bottom left of the IDE is used to open serial connection (serial monitor can be opened using a tab just to its right), and scripts can be run and stopped using the button below this button (Figure 1, red and blue circles).

The IDE allows you to view a stream of the camera output (Figure 1, top right), which can be handy for development. However, keeping the stream enabled severely impacts the performance of the bot, in many cases halving the video frames-per-second. Therefore, we suggest you disable the stream when you want to test the maximal performance of your bot by using the “Disable” toggle at the top right of the frame buffer (Figure 1).

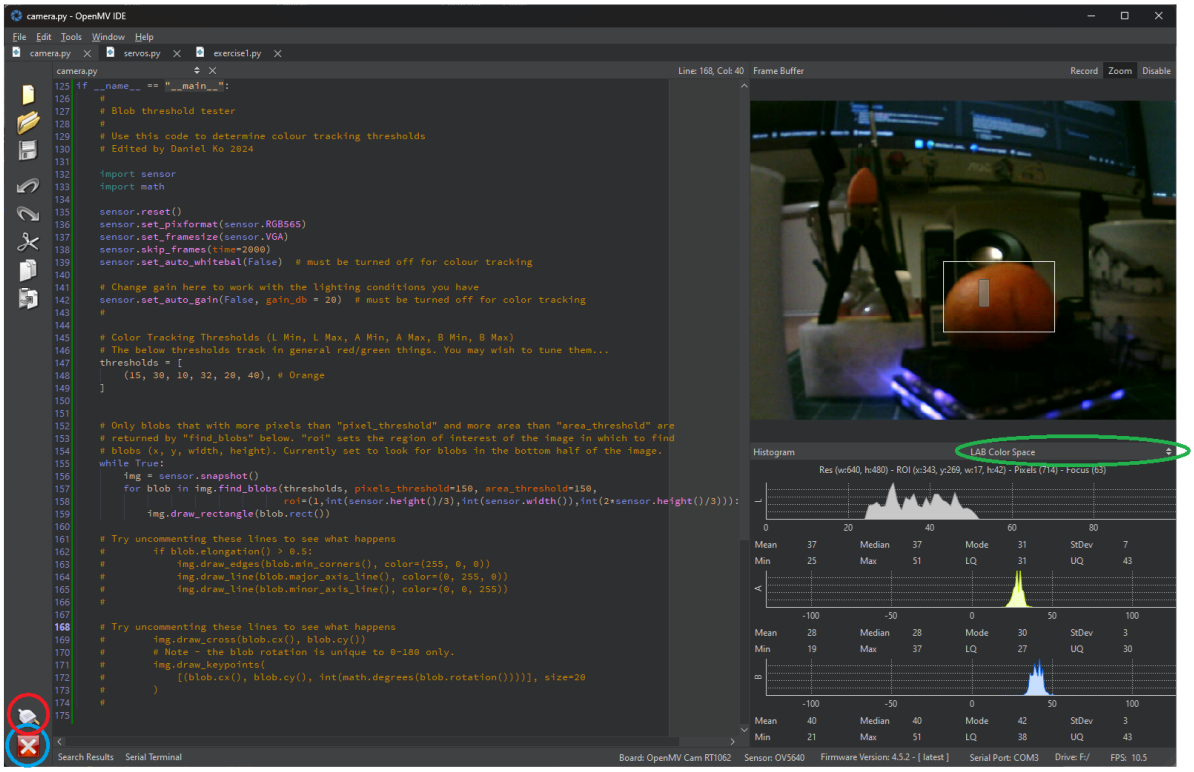

Figure 1: OpenMV IDE running colour tracking example. The buttons at the bottom left corner of the IDE are used to connect/disconnect to/from the OpenMV (red circle) and run/stop the current script open on the IDE (blue circle). When camera.py is run as a script, the frame buffer will be streamed from the OpenMV for viewing. The histogram window has basic statistics printed out just below the histograms for each colour channel that you can use to determine thresholds.

Colour detection threshold tuning

You can run ***camera.py*** as a script using OpenMV IDE to tune colour detection thresholds (check the end of the file to set thresholds). Colour tracking thresholds are set using minimum and maximum values in LAB space (Figure 2). The colour histograms below the frame buffer can be set to use LAB space using a dropdown (Figure 1, green oval), and focused on specific regions of interest by clicking and dragging rectangles in the image buffer (Figure 1, grey filled rectangle). Upon re-running the script, any detected colours will be marked in the frame buffer with a white rectangular boundary (Figure 1).

You can also use Tools->Machine Vision->Threshold editor to adjust the thresholds using sliders in OpenMV IDE.

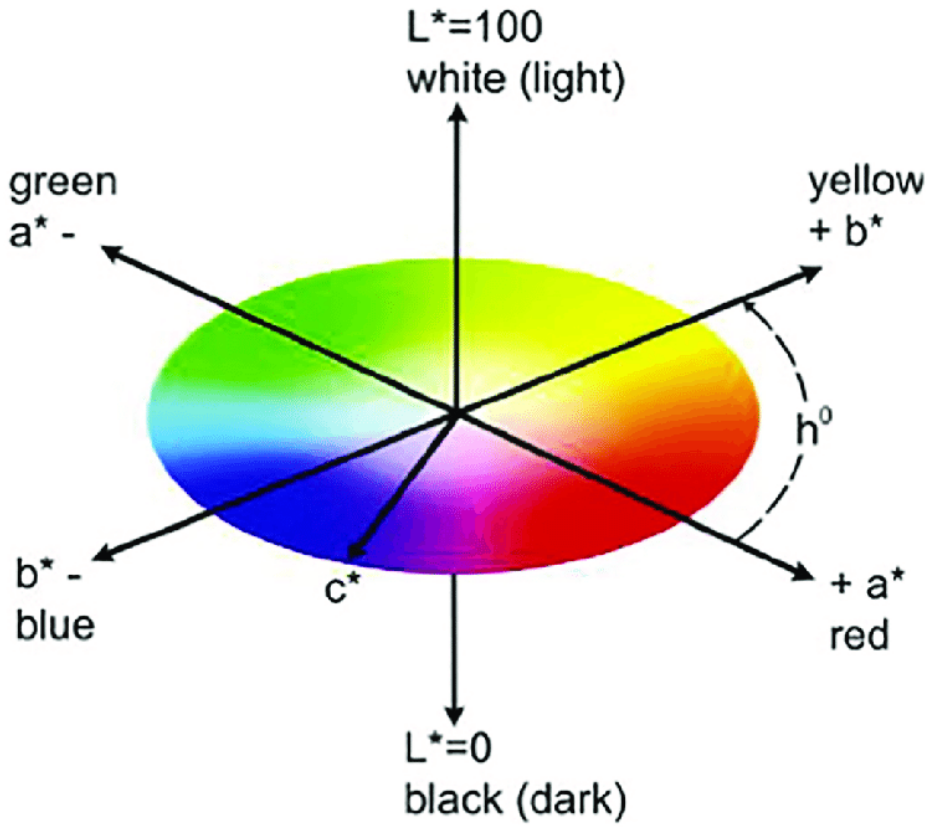

Figure 2 LAB Colour space. Reproduced from Ebeneezar et al 2020 (10.1016/j.aquaculture.2020.735728)

Running scripts over wifi

For cable-free testing, you can execute scripts on the OpenMV_bot over wifi. You can save ***MV_remote_exec.py*** as ***main.py*** onto the bot's SD card to make sure the bot runs it on startup, regardless of if it is connected to a computer or not. This sets up the OpenMV to act as a wifi access point for your compuer to connect to. From the next power cycle, the bot will listen over wifi for scripts to be sent over and executed. You should modify ***MV_remote_exec.py lines 11-12*** to set the SSID and password of your access point so you do not accidentally connect to someone else's bot.

Once your computer is connected to the access point, you need to find the IP address of the bot (e.g. ipconfig on command line) and enter it into ***send_script.py line 19***. Additionally, you need to enter the path of the script file on your computer that you want to execute (***line 21***). Finally, run ***send_script.py*** on your device. You will not be able to use OpenMV IDE for this purpose so will need another way to run python scripts (e.g. terminal with python installed, Visual Studio Code).

MAKE SURE TO NOT HAVE ANY INFINITE LOOPS IN THE SCRIPT YOUR SEND OVER!!! YOU WILL NOT BE ABLE TO STOP EXECUTION VIA SOFTWARE SO YOU WILL HAVE TO CATCH YOUR ROBOT AND HARDWARE RESET!!!

Hardware usage notes

General notes

- Please treat the OpenMV_bot with care, covering the optics when not in use and making sure the SD card isn't lost/damaged. Use a small box to transport the bot is a great way to avoid accidents.
- Turn the OpenMV off (see below) when not in use to conserve battery – you don't want to run out of battery in the middle of an assignment!
- You should not have to take apart the bot at all throughout the course, but if things break, do let one of the GTAs know so we can troubleshoot and hand out spares.

Power cycling

The OpenMV_bot can be powered by either the USB cable, battery or both. To turn off your robot make sure the USB cable is disconnected and the switch on the back is off. **This is an effective way of resetting your robot in the case of an infinite loop (which should be avoided regardless).** When the robot is not in use, please make sure the battery is turned off, so it is not being drained to empty unnecessarily.

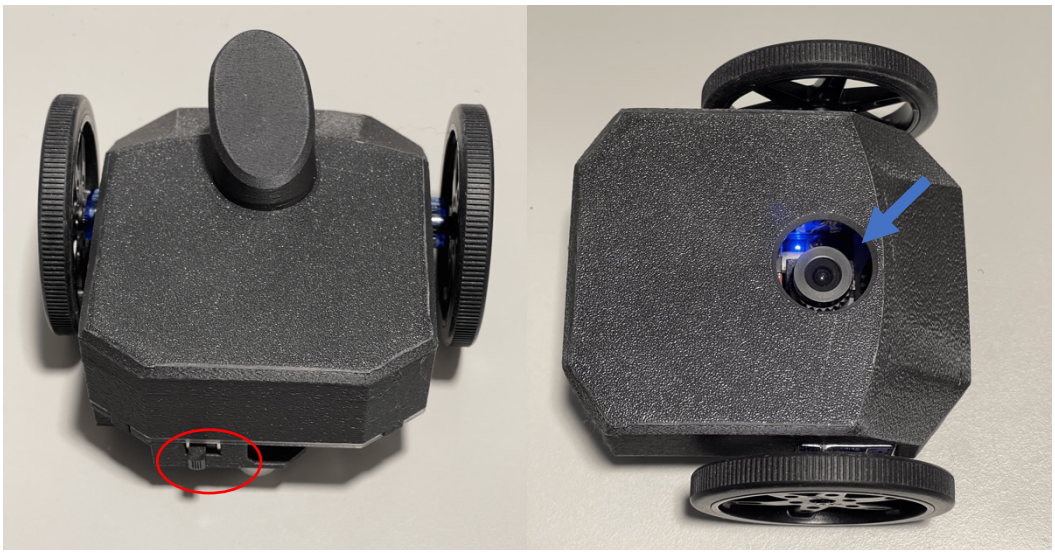

Figure 3: OpenMV from the top. The power switch is in the red circle. The camera focus can be adjusted by screwing/unscrewing the camera lens (blue arrow, see page 5).

Changing the camera focus

The camera focus can be adjusted by first removing the top case, then loosening the knurled nut on the lens thread (Figure 3) and then turning the camera lens (Figure 3, blue arrow) until the image comes into desired focus. Finally, the knurled nut should be re-tightened to secure the focus position. **Please be careful not to damage the lens!**

Servo zero-point offset tuning (important)

Each of your wheels may have different zero-points, meaning even when the drive setting is set to 0, the wheel may move slowly in either the backwards or forwards direction. To fix this, you can adjust the zero-point offsets for each wheel in ***servos.py lines 16-17*** incrementally (try +-0.05 steps) until both wheels stay still at drive setting of 0. Running ***servos.py*** as a script may help you tune these parameters.

Similarly, you should set ***servos.py line 15*** to centre the camera pan servo when pan angle is set to zero.

Wheel speed tuning instructions (optional)

You may notice that the wheel speed does not scale linearly between -1 and 1 drive setting in the control code. If you would like to have better control over the speed of your OpenMV_bot, you may choose to perform wheel speed tuning experiments to model the drive setting-actual speed curve (e.g. sinusoidal, sigmoid). You will need a printed turning pad (on Blackboard) to perform these experiments.

Instructions:

1. Place the bot on the turning pad with one of the wheels placed in the centre and aligned with the 0° line. You will be testing the dynamics of the other wheel (the "test wheel") (Figure 4, left).
2. Test how many degrees the bot rotates around the turning pad at several different drive settings for the test wheel (Figure 4, right). Note these values down. We suggest you sample the lower drive settings more densely. For example, you may test with drive settings of -1,-0.7, -0.4, -0.2, -0.15, -0.1, -0.05 for backwards drive.
3. Plot the drive settings on the x-axis and the angular distance values on the y-axis. What relationship do you see? Can you fit a function to it? Maybe you need to fit four separate functions for each of left wheel forwards/backwards and right wheel forwards/backwards.
4. Code the drive setting-actual speed curve onto your bot as a class method of *Servo* in ***servos.py***. You can modify the default drive methods to call on this new method.

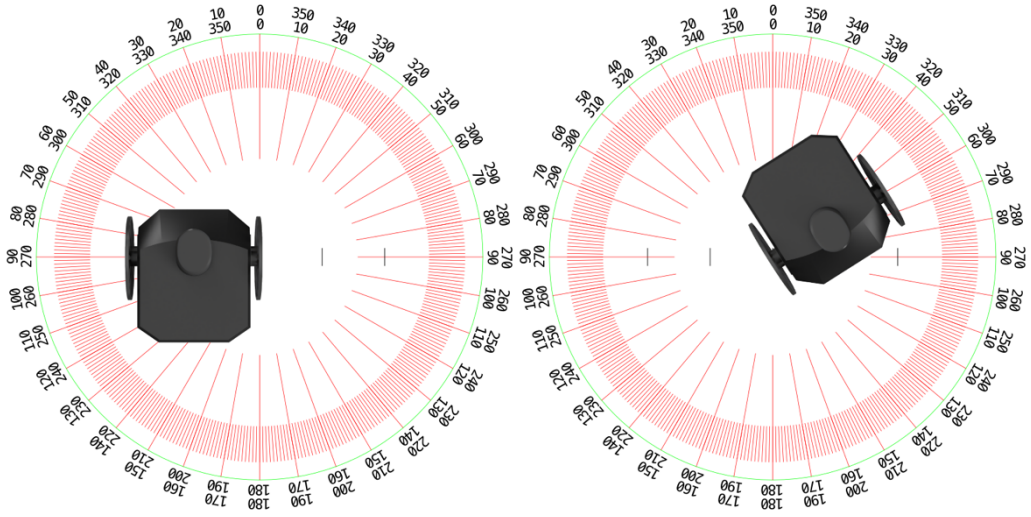

Figure 4: Using a turning pad to tune the drive settings of the left wheel. (left) the bot is placed with the non-test wheel in line with the mark in the centre of the circle to begin. (right) the bot has ended up with the non-test wheel in line with the 140° mark after driving the test wheel forwards. Write this value down. Note that the bot may rotate more than 360° for higher drive settings.

Common errors and fixes

Trying to run a script gives me Err19!

Make sure the servo shield on the OpenMV has not become unconnected from the main board – be careful when unplugging/replugging the USB cable.

I am editing my modules but the changes aren't reflected when I run the code!

The modules are being cached by OpenMV for faster compiling, so any updates to modules are not recompiled every time you run a script. Try moving all your module and script files onto the base directory of the SD card and running them there. If that does not work, try a hardware reset by power cycling the OpenMV.

I am getting a missing module error, even when the module exists!

Try moving all your module and script files onto the base directory of the SD card and running them there. If that doesn't work, try a hardware reset.

My robot performs differently depending on connection type!

Disable the image buffer when testing on USB connection. See page 2.

The OpenMV camera is out of focus!

You can focus the camera manually. See page 5.