GNSS + VIO instructions

Required hardware

- u-blox C099-F9P + ANN-MB-00
- Luxonis OAK-D
- Laptop running Linux

GNSS antenna should be connected rigidly to the OAK-D. However the USB3 cable from OAK-D will interfere with the GNSS, so the cables should be kept separate and as far apart as possible, preferably shielded with metal.

If the distance from GNSS antenna to OAK-D device is large (>1m), you can improve accuracy by measuring it and giving the translation to Spectacular AI DepthAI using `gnssToImu` parameter as seen in `vio_gnss.py` example.

u-blox one time setup

Clone u-blox-capture repository.

```
git clone https://github.com/SpectacularAI/u-blox-capture --recurse-submodules
```

Configuring u-blox C099-F9P application board

Let's configure the ublox first to output high precision GNSS locations and nothing else to ensure minimal latency. Find out the `DEVICE` name and run the following command to apply the settings, these will reset when the device is rebooted. Most likely the device is `/dev/ttyACM0` where 0 may be a higher number.

```
python ubx_configurator.py DEVICE example/high_precision_gps_only.cfg
```

Once you've tested that the settings work, you can make them permanent by using -flash flag. After this they will not be reset on reboot. Alternatively you can run the settings command above every time you wish to use the device.

```
python ubx_configurator.py DEVICE -flash example/high_precision_gps_only.cfg
```

Building RTKLIB

Next we must build RTKLIB which is used to pipe the ground station signal from the internet to the u-blox device.

```
cd RTKLIB/app/str2str/gcc/
make
```

Running and recording GNSS + VIO

1. Pipe RTK signal to device

- RTK information from your provider: `USER`, `PASS`, `IP`, `PORT`, `MOUNTPOINT`
- `LAT/LON` Rough estimate of current position
- `DEVICE` device name, for example: ttyACM0

```
./RTKLIB/app/str2str/gcc/str2str -in ntrip://USER:PASS@IP:PORT/MOUNTPOINT \
      -p LAT LON 0.0 -n 250 -out serial://DEVICE:460800:8:n:1
```

Leave this running in the background. It may take minutes to find all the satellites and reach maximum accuracy.

u-blox device should now show a *blinking blue light* (connected to GPS) and *solid yellow light* next to it (connected to RTK).

2. Testing that GNSS works

Next we'll ensure that GNSS is working properly, print the output with

```
python ubx_stdout.py DEVICE —-json
```

You should see JSON formatted GNSS positions and ideally the "accuracy" should be below one meter (lower is better).

3. Running GNSS-VIO

If you haven't tested Spectacular AI sdk-examples yet, it's recommended that you do so before proceeding further. You should at least test that the `vio_visu.py` works as expected.

Next we'll pipe the GNSS positions to `vio_gnss.py` which combines it with vio.

```
python ubx_stdout.py DEVICE —-json | python vio_gnss.py
```

You will need to move about 30 meters, so the GNSS trajectory is long enough to align with vio. Once this happens, you will see latitude and longitude coordinates instead of the relative position.

4. Recording

To record the session, add a recording folder as an argument to `vio_gnss.py`. It will overwrite existing data when run again, so adding a timestamp to the folder name is recommended.

```
python ubx_stdout.py DEVICE —-json | python vio_gnss.py ./data/
```