Implement a VIO algorithm for Crazyflie

Riccardo Gattoni
Tommaso Furlani

Goals

- Reading data from IMU sensors
- Collecting a sequence of frames from the monocamera
- Temporally synchronizing the two types of data
- Organizing the collected data in a specific path
- Running Kimera_VIO to visualize the trajectory

Main requirements

Hardware

- Crazyflie 2.1
- Crazyradio 2.0
- Ai-deck with monocamera HIMAX HM01B0
- Jtag-gap8

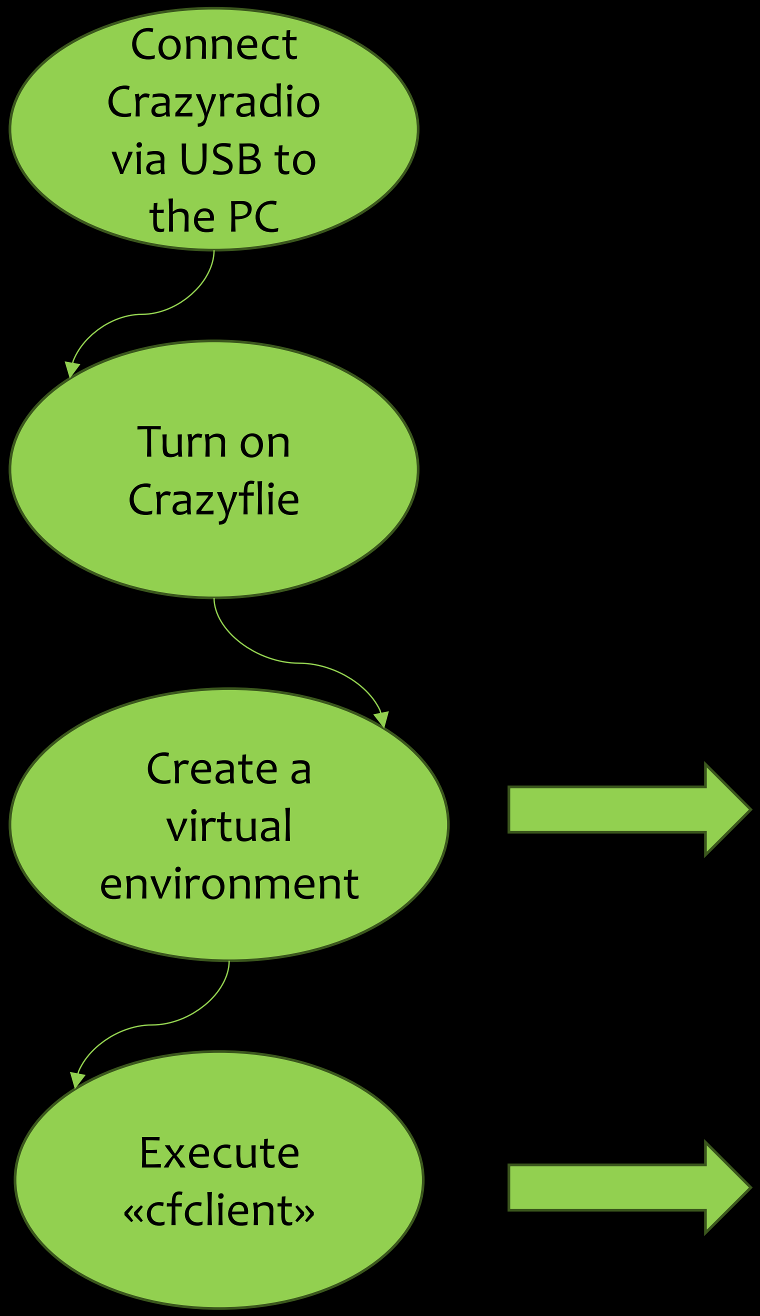
Software

- Ubuntu 24.04
- Cfclient
- Kimera-VIO algorithm

Crazyflie 2.1

Open-source platform with the possibility of additional support decks, such as flow_deck and ai_deck. Ability to control the device through Python client on the PC.

Cfclient implementation



Connect Crazyradio via USB to the PC

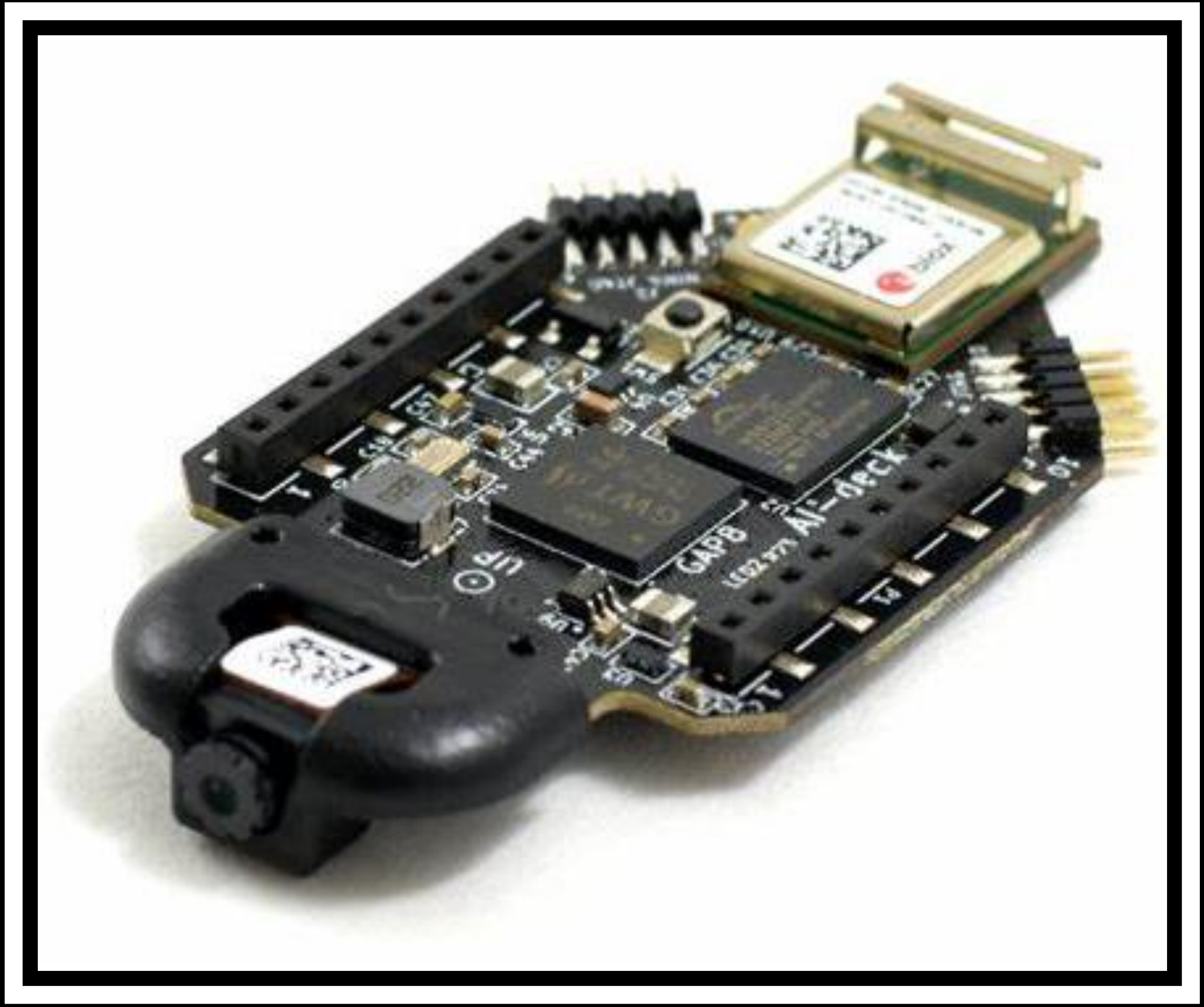
Turn on Crazyflie

Create a virtual environment

```
$python3 -m venv cfclient-env
```

Execute «cfclient»

```
$cfclient
```



AI-DECK

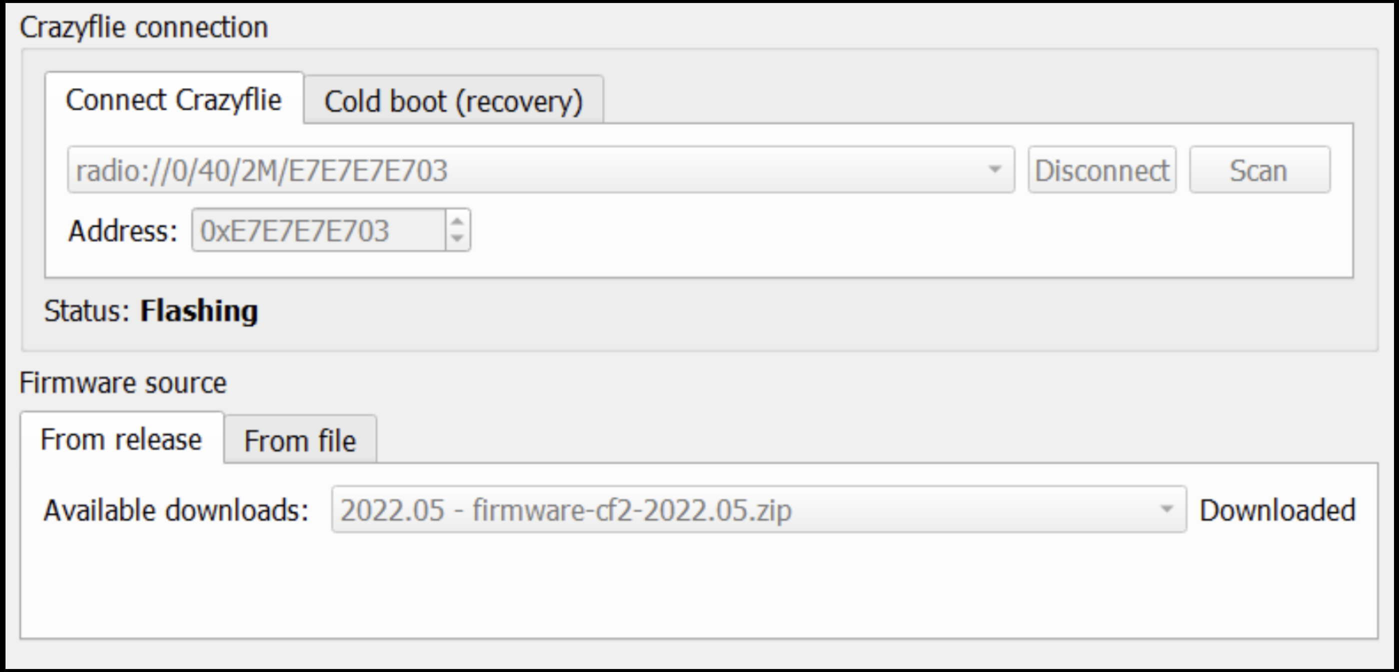
- Support deck attached on top of Crazyflie.

- **GAP8 processor** programmed with JTAG, enabling AI algorithm execution and signal processing with low energy consumption.

- **ESP32 microcontroller** provides WiFi connectivity to ensure image streaming.

AI-DECK implementation

ESP32 → Flashing the firmware for the microcontroller → ESP32: I (910) SYS: Initialized



Crazyflie connection

radio://0/40/2M/E7E7E7E703

Address: 0xE7E7E7E703

Status: **Flashing**

Firmware source

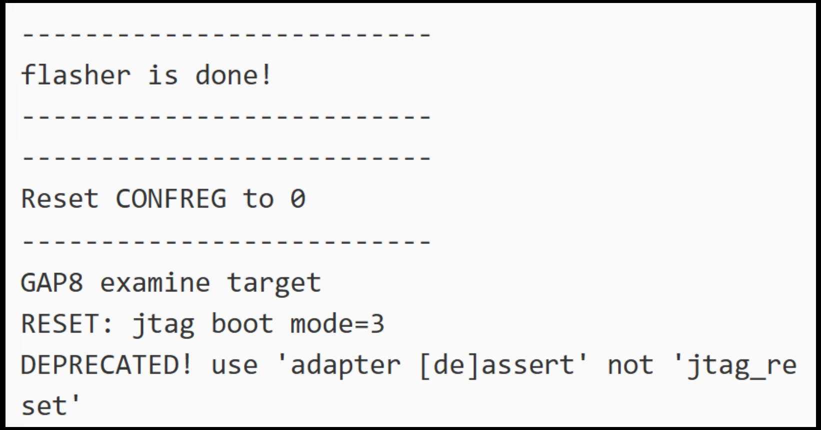
Available downloads: 2022.05 - firmware-cf2-2022.05.zip Downloaded

GAP8 →

Flashing the bootloader on GAP8 via JTAG :

```
$docker run -rm -it -v $PWD:/module/ --device
/dev/ttyUSB0 -privileged -P bitcraze/aideck
/bin/bash -c ‘export
GAPY_OPENOCD_CABLE=interface/ftdi/olimex-arm-
usb-ocd-h.cfg; source /gap_sdk/configs/ai_deck.sh;
cd /module/; make all image flash’
```

→

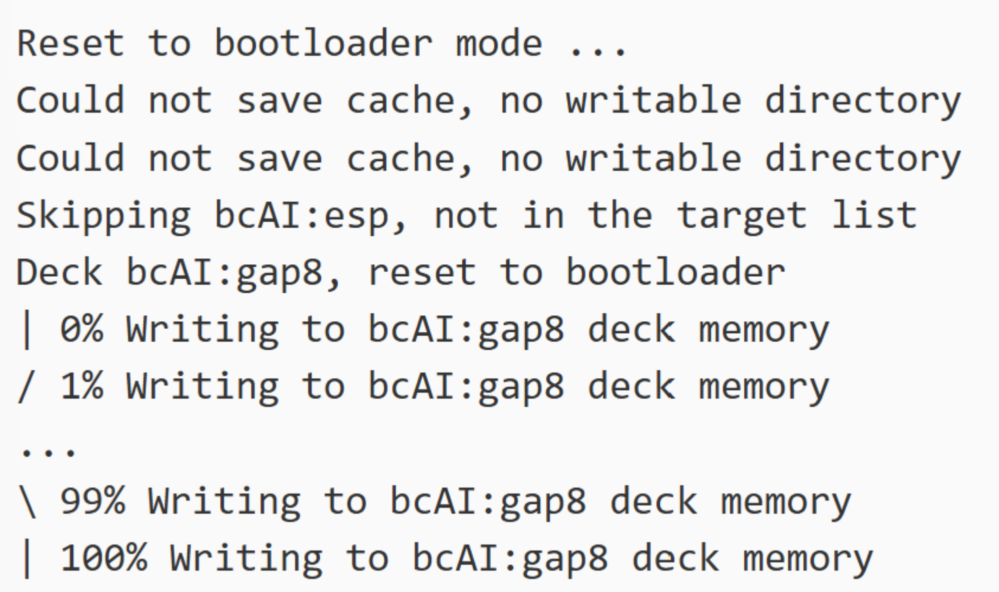


```
--------------------------
flasher is done!
--------------------------
--------------------------
Reset CONFREG to 0
--------------------------
GAP8 examine target
RESET: jtag boot mode=3
DEPRECATED! use 'adapter [de]assert' not 'jtag_re
set'
```

Flash wifi-example

- Connect the PC to the WiFi of the ESP32 on the Ai-deck to enable image streaming

```
$cfloader flash aideck_gap8_wifi_img_streamer_with_ap.bin deck-bcAI:gap8-fw -w radio://0/84/2M/E7E7E7E7EB
```

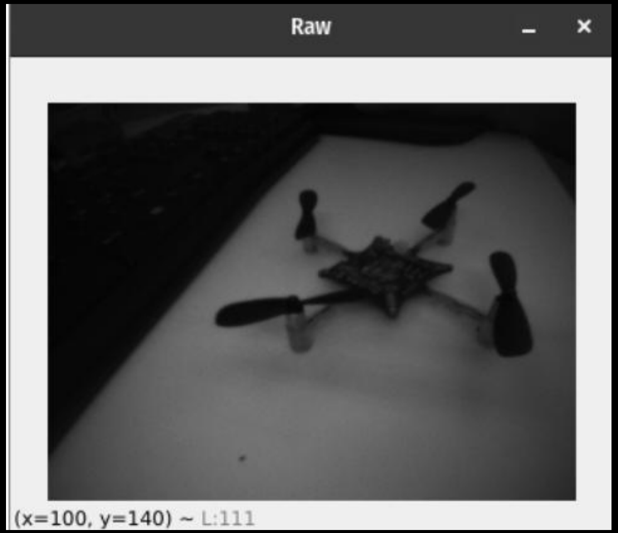


```
Reset to bootloader mode ...
Could not save cache, no writable directory
Could not save cache, no writable directory
Skipping bcAI:esp, not in the target list
Deck bcAI:gap8, reset to bootloader
| 0% Writing to bcAI:gap8 deck memory
/ 1% Writing to bcAI:gap8 deck memory
...
\ 99% Writing to bcAI:gap8 deck memory
| 100% Writing to bcAI:gap8 deck memory
```

Flash wifi-example

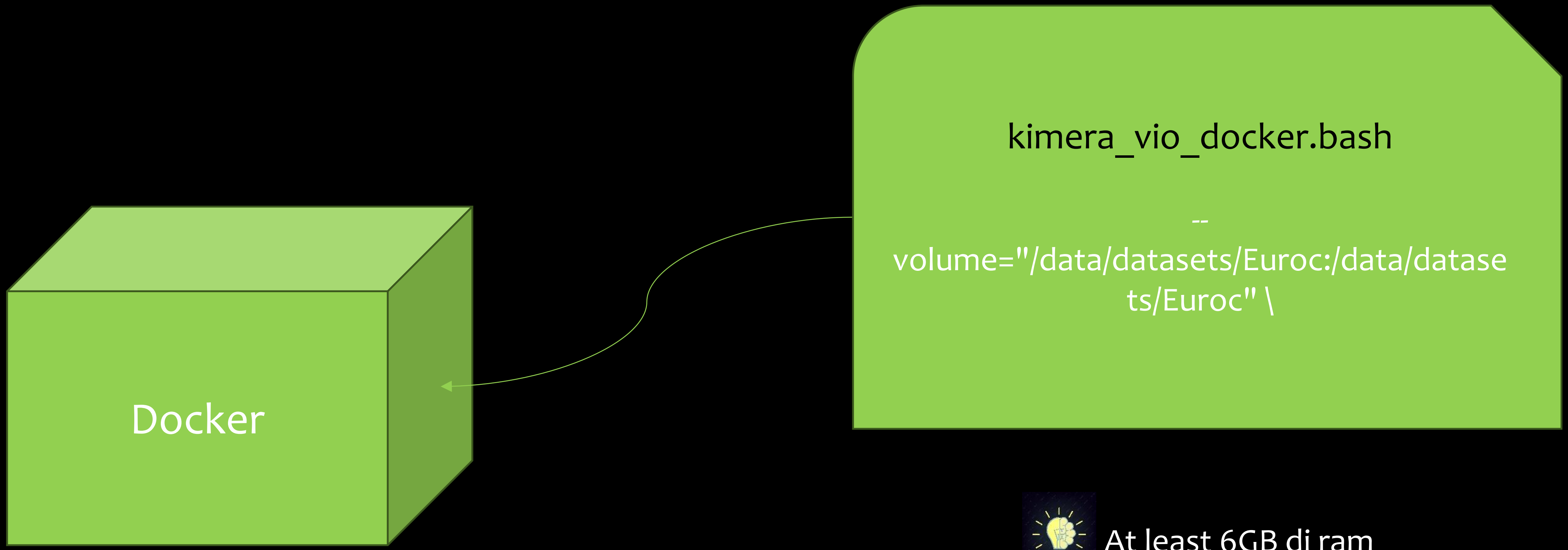
- Now it is possible to find the WiFi network 'Wifi streaming example' and connect the PC.
- Then, execute from within the previously described virtual environment.

```
$python3 opencv-viewer.py
```



```
(x=100, y=140) ~ L:111
```

Kimera-VIO installation



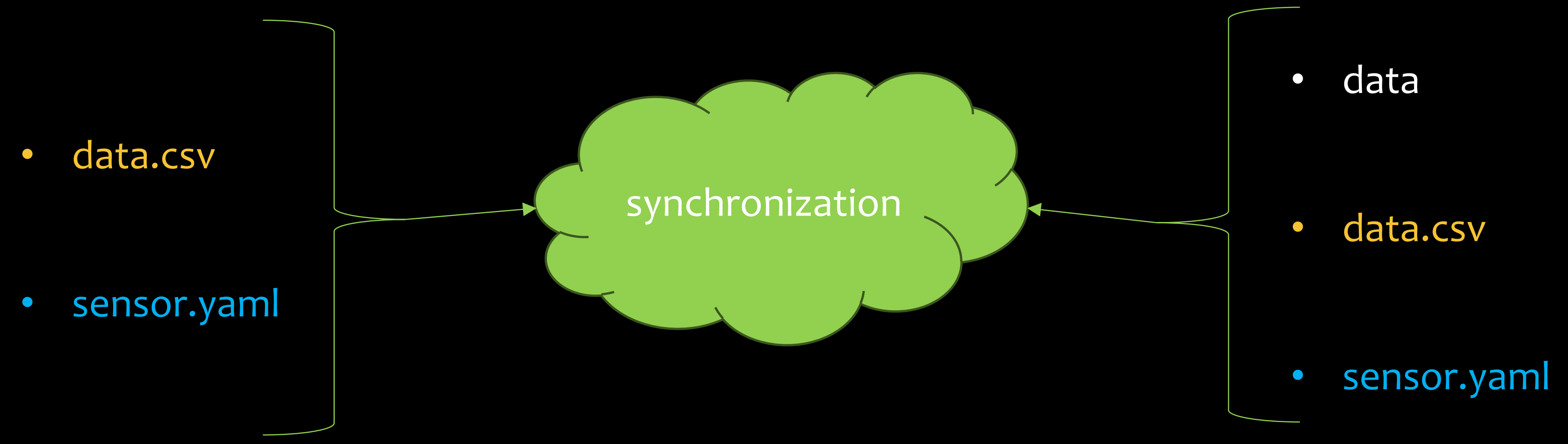
Docker

kimera_vio_docker.bash

```
--volume="/data/datasets/Euroc:/data/datasets/Euroc" \
```

At least 6GB di ram

How to collect data

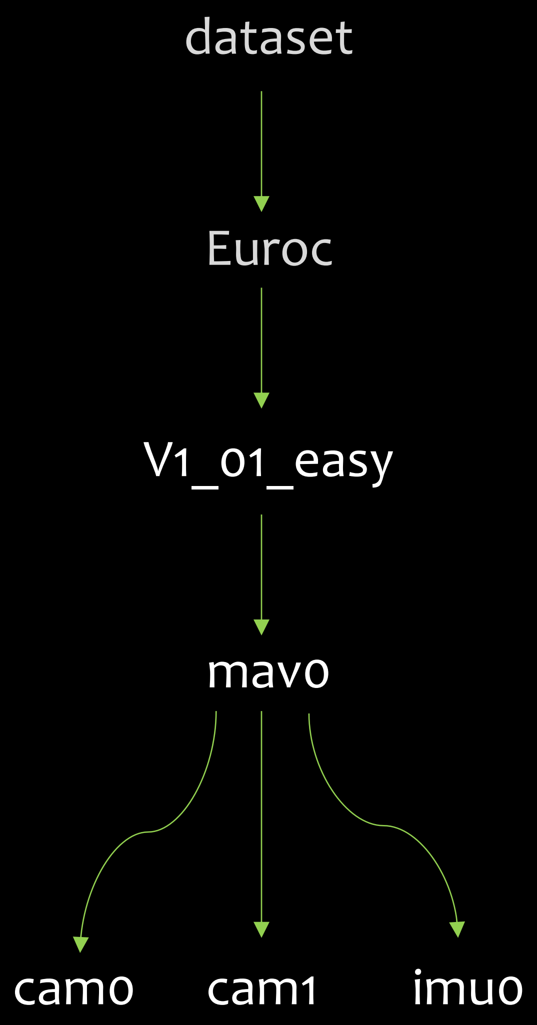


- data.csv
- sensor.yaml

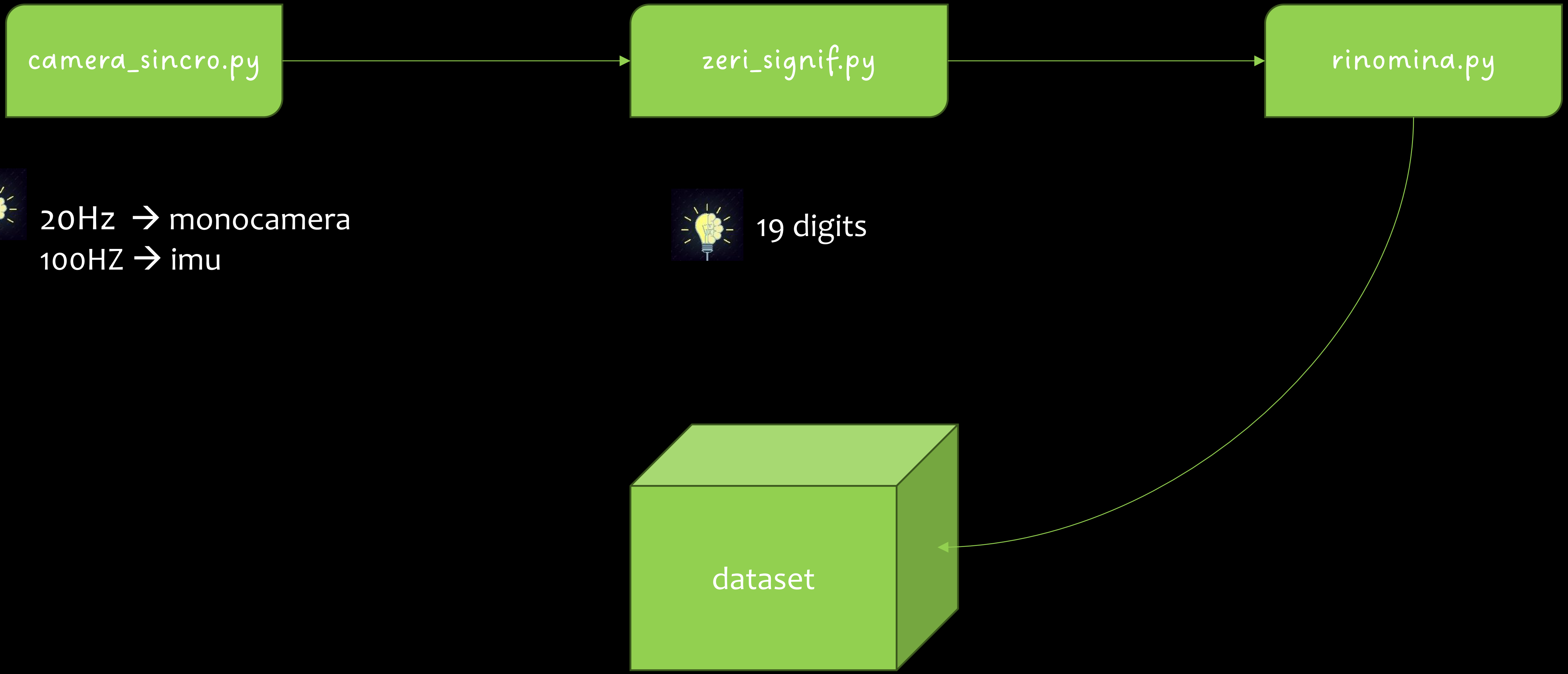
synchronization

- data
- data.csv
- sensor.yaml

Dataset organization scheme for Kimera-VIO



Data synchronization



camera_sincro.py

zeri_signif.py

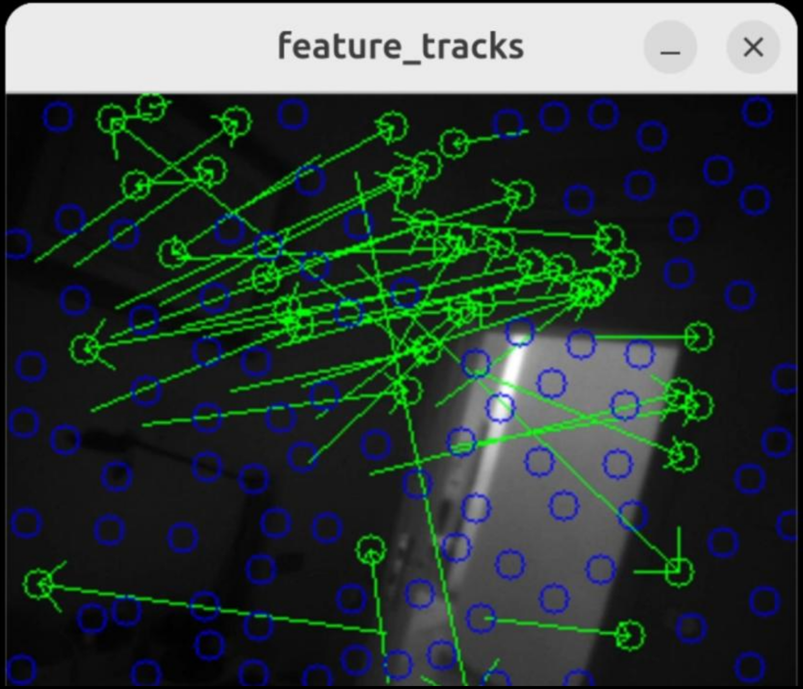
rinomina.py

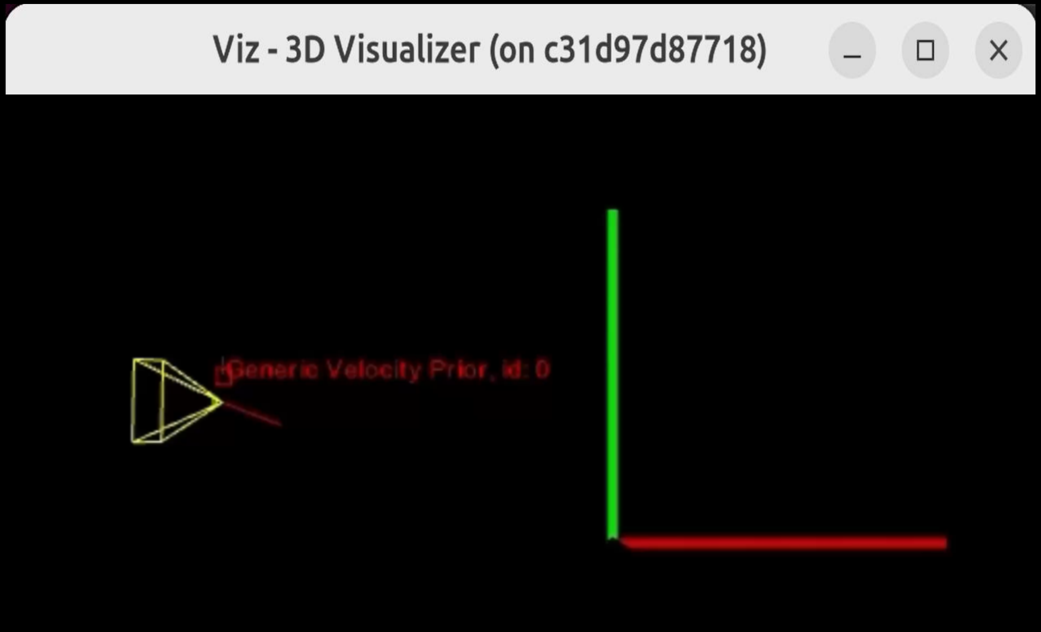
dataset

20Hz → monocamera
100HZ → imu

19 digits

Simulation


feature_tracks


Viz - 3D Visualizer (on c31d97d87718)
Generic Velocity Prior, id: 0

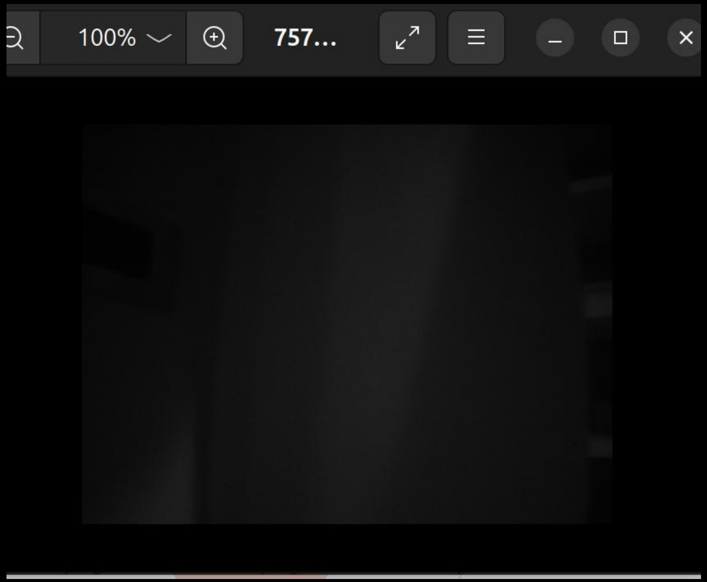
Issues

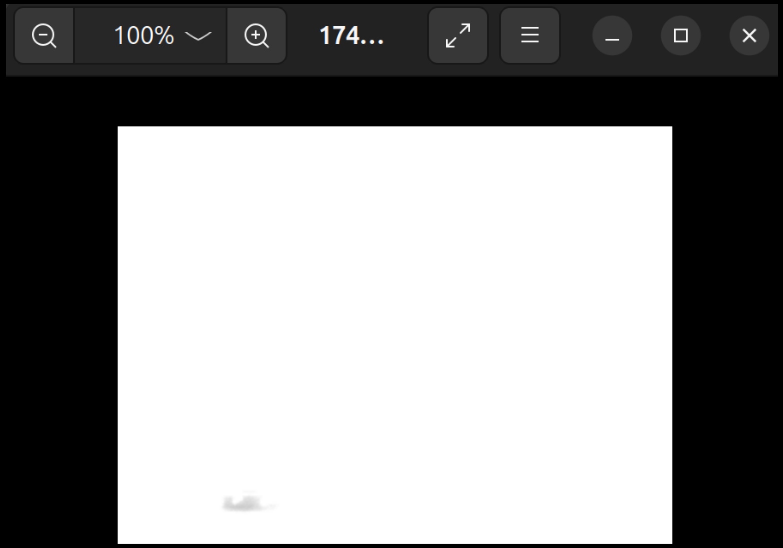
- Frame size incompatibility (244 x 324)

sensor.yaml

- Multiple simulations

  - good lighting
  - low noise
  - frames





- Study of synchronization and research on the frame rate of the monocamera.

Future implementations

- Creating a video with even better quality

- Research on more detailed synchronization

All documentation can be found at the following link:

rgattoni/VIO-project-with-Kimera_VIO: Implement a VIO algorithm for Crazyflie

Thank you for your attention