Projet COBRA équipe localisation

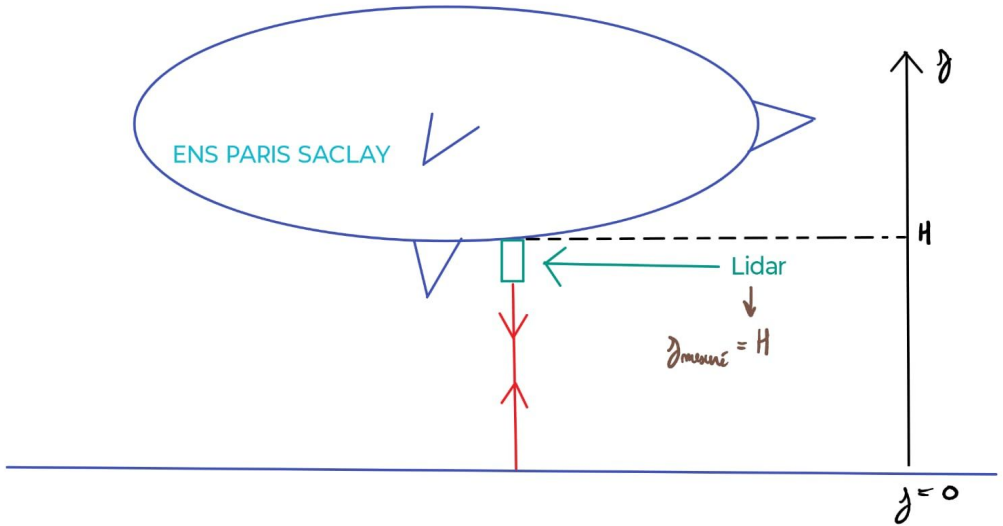
Présentation et avancement

Sommaire

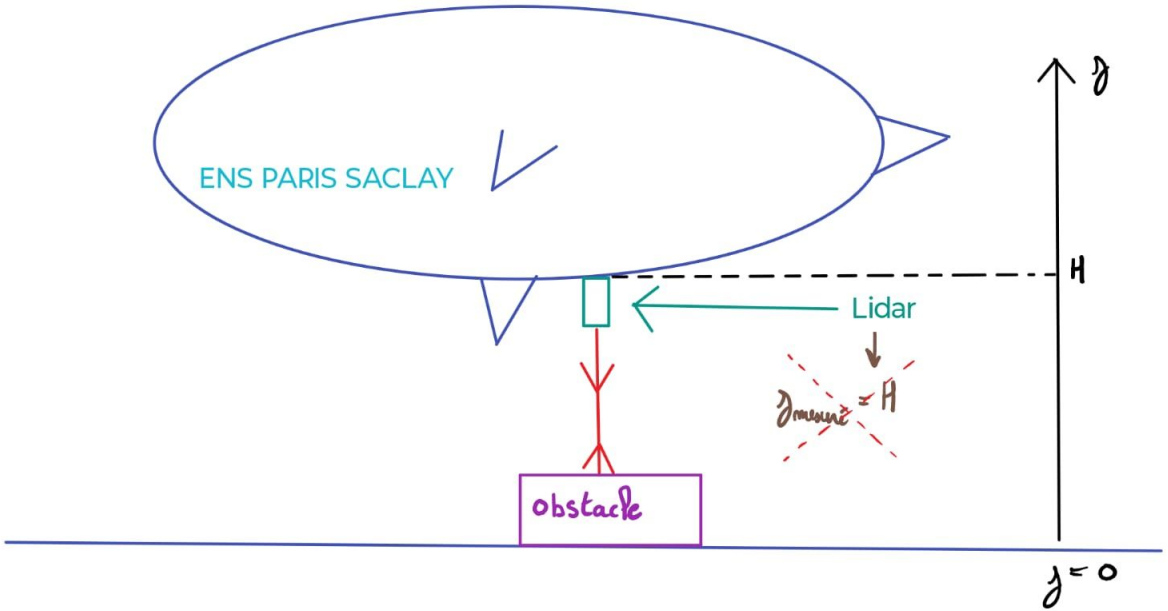
I-Capteur de pression
II-Nouvelle méthode de localisation
III-GPS indoor
IV-Vslam
V-Centrale inertielle
VI-Apriltags

Choix et création d’une bibliothèque d’un capteur de pression

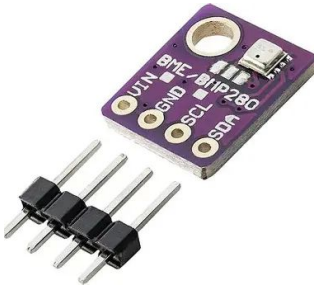
Intérêt d'un capteur de pression



Intérêt d'un capteur de pression

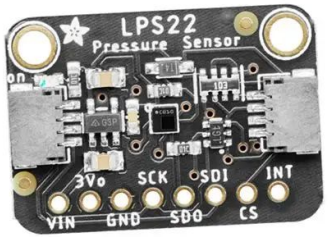


1.Etude théorique



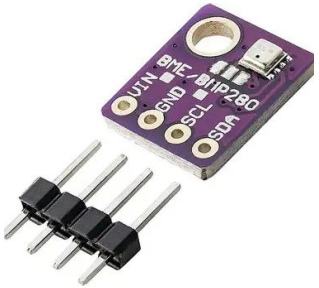
BME280

ICP-10111



LPS22HB

1.Choix du capteur



BME 280

Resolution of pressure output data	R_P	Highest oversampling		0.18		Pa

$$\frac{\partial P}{\partial z} = -\rho g \quad \text{soit} \quad \Delta z = \frac{\Delta P}{\rho g}$$

résolution d'environ 1.5cm

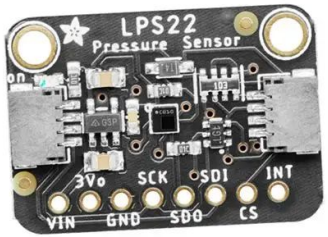
1.Etude théorique

ICP-10111

Resolution	Maximum range	0.01	Pa	

résolution d’environ 0.8 mm

1.Etude théorique



LPS22HB

P_{sens}	Pressure sensitivity			4096		LSB/ hPa

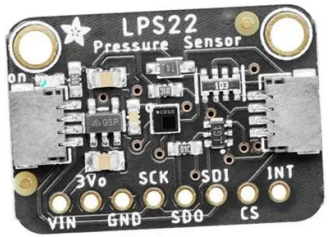
$$s = \frac{\Delta z}{4096}$$

une variation d'1hPa équivaut à une variation de 8.5m
=> résolution de 0.2cm

1.Etude théorique

BME280

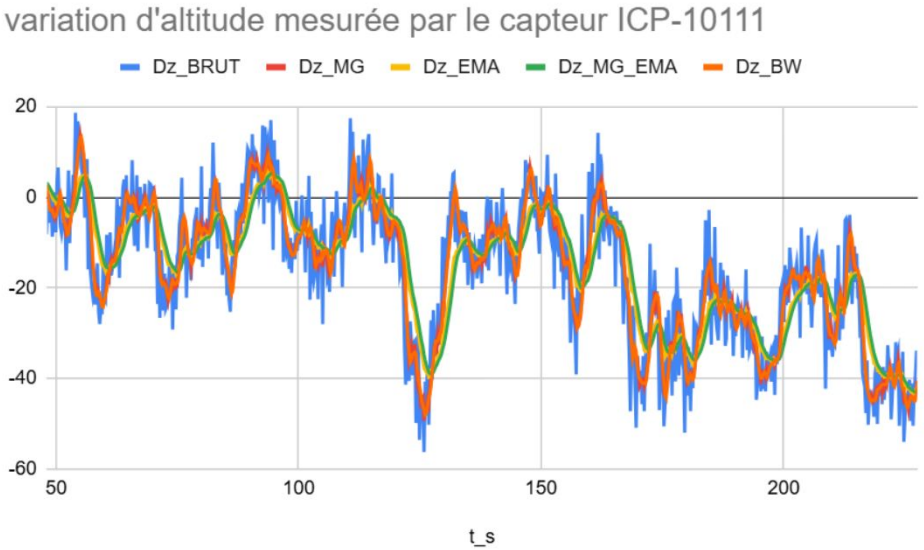
ICP-10111



LPS22HB

2. Test

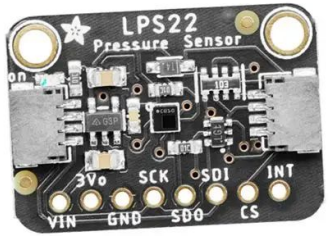
ICP-10111



2. Test

BME280

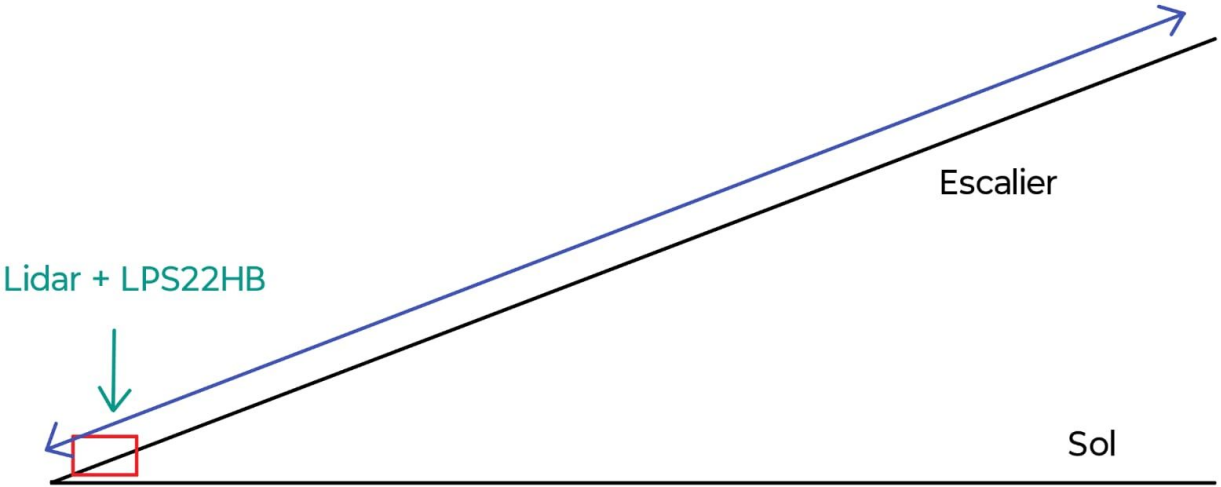
ICP-10111



LPS22HB

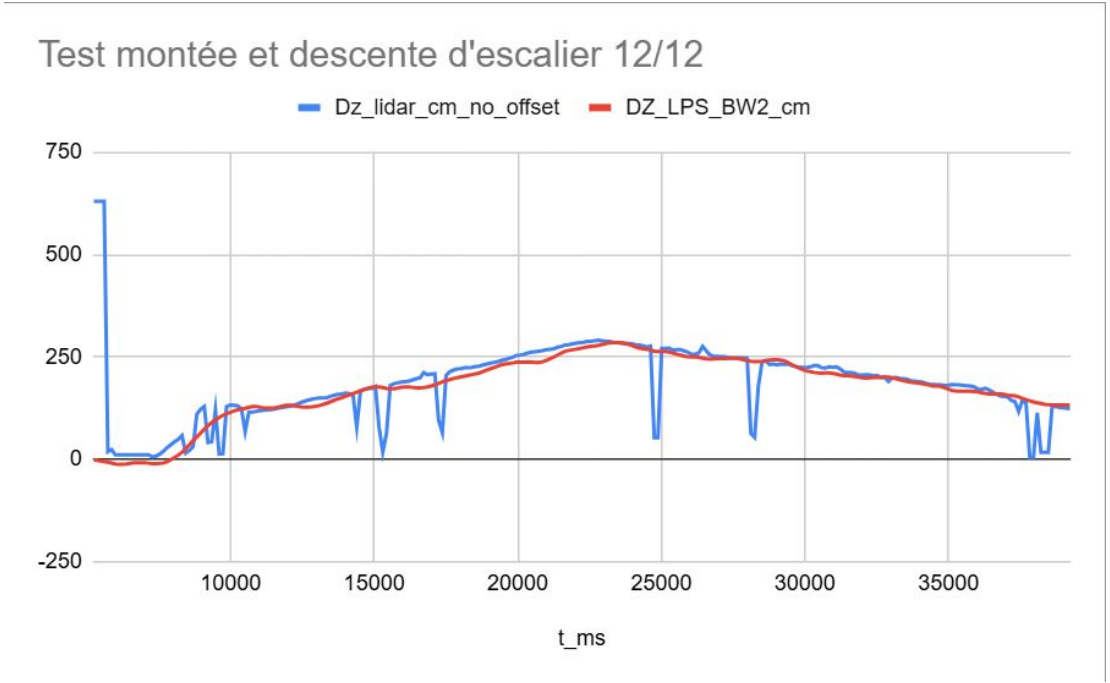
2.Test (de l'escalier)

LPS22HB



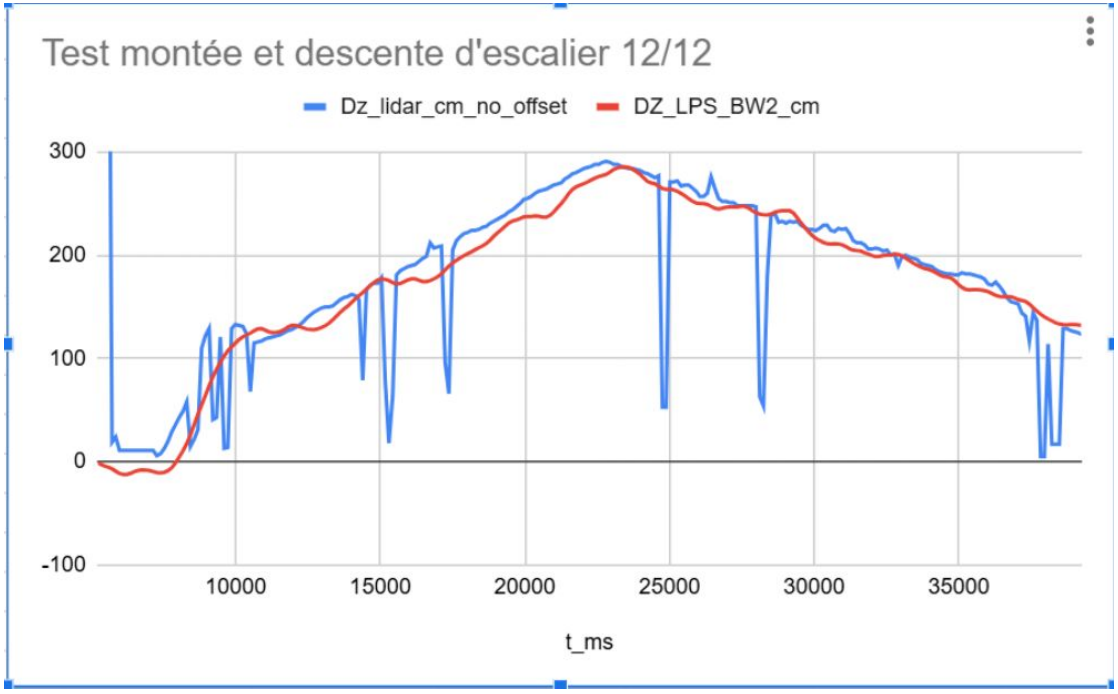
2.TEST (de l'escalier)

LPS22HB



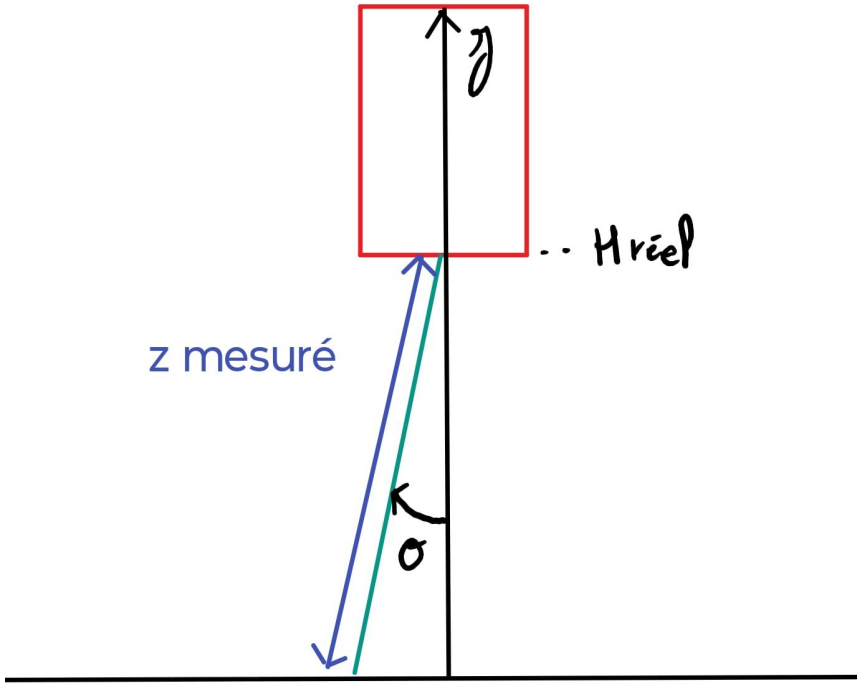
2.TEST (de l'escalier)

LPS22HB

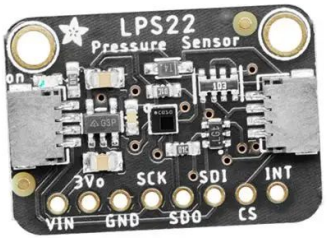


2.TEST (de l'escalier)

erreur possible due au Lidar



3.Choix du capteur



LPS22HB

4.Création d'une bibliothèque

```python
from smbus2 import  SMBus

class LPS22:
    DEFAULT_ADDRESS = 0x5C

    CTRL_REG1 = 0x10 #addres of CTRL_REG1
    CTRL_REG2 = 0x11 #adress of CTRL_REG2

    ODR_RATE = { #rate associate to their binary code
    0:  0b000,  # power-down
    1:  0b001,  # 1 Hz
    10: 0b010,  # 10 Hz
    25: 0b011,  # 25 Hz
    50: 0b100,  # 50 Hz
    75: 0b101,  # 75 Hz
    }

    DEVICE_BANDWIDTH ={ #efficient only if EN_LPFP=1
    2:  0b00,
    9:  0b10,
    20: 0b11,
    }

    NOT_ODR_MASK = 0b10001111
    NOT_LPF_MASK= 0b11110011
    BDU_MASK = 0b11111101
    IF_ADD_INC_MASK = 0B00010000
    NOT_FIFO_MASK = 0b10111111
    SWEET_RESET_MASK = 0b00000100
```

```python
def read_registers(self, start_reg, length):#permit to read numerous registers

    #start_reg = address of te first register you want to read
    #length = number of the register you want to read

    return self.bus.read_i2c_block_data(self.address,start_reg, length)

def write_register (self, reg, data):

    #reg = adress of the register where you want to write
    #data= data that you want to write

    self.bus.write_byte_data(self.address, reg, data & 0xFF)
```

4.Création d'une bibliothèque

```
def low_pass_filter(self, bandwidth: int):

    #bandwidth = n, LPF = ODR/n, for n = 2,9,20


    #note à Monsieur Juton, j'ai lu qu'il fallait mieux séparer les bits "ENABLE" des bits de config, est-ce vraie ? Si oui pourquoi ?

    if bandwidth not in self.DEVICE_BANDWIDTH:
        raise RuntimeError("PLZ USE A BANDWIDTH THAT EXIST")
    lpf_bits = self.DEVICE_BANDWIDTH[bandwidth]
    data = self.read_register(self.CTRL_REG1)
    data &= self.NOT_LPF_MASK ##EN_LPF AND LPF_CFG cleared
    data |= lpf_bits << 2
    self.write_register(self.CTRL_REG1, data)
```

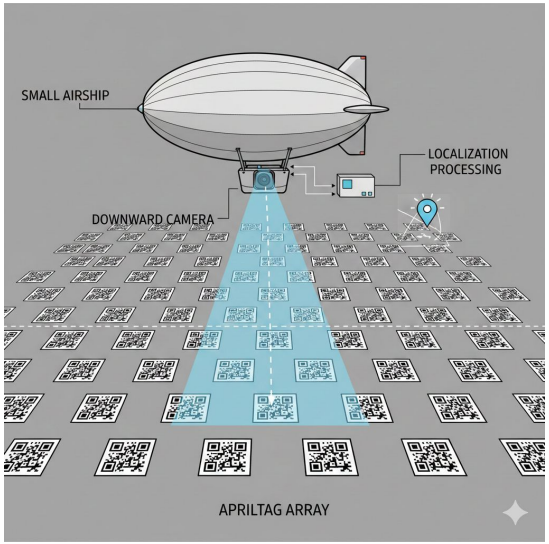
5.Conclusion

-Rien ne prouve que cette solution sera efficace à 100%
-Le capteur de pression ne semble pas assez précis pour être l'unique instrument de mesure de l'altitude
-Permet d'améliorer la robustesse du système (peu sensible aux tangages, roulis, lacets, obstacles etc…)

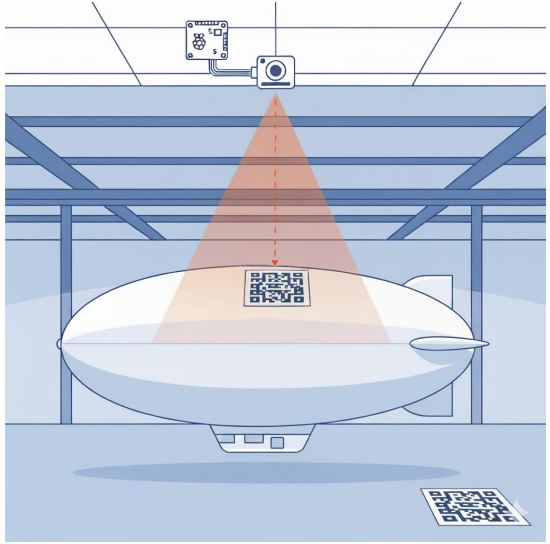
Nouvelle méthode de localisation

Implémentation d'une nouvelle méthode de localisation du dirigeable

Ancienne méthode: Caméra embarquée sur le dirigeable et détection de tags sur le sol pour se localiser.



Nouvelle approche: Caméra fixe non embarquée qui filme le dirigeable depuis le dessus, ce dernier ayant un tag sur son dos. L'environnement étant muni d'un repère.



Avantages

-Moins de tag au sol→plus de portée: on peut multiplier les angles de caméra (3 ou 4 repères au sol pour couvrir la moitié de l'atrium).

-Pas besoin d'une installation millimétrée de 20 tags sur le sol

-On garde la coordonnée du repère en mémoire (fixe) et donc si quelqu'un passe dessus on ne perd pas la loc

-Portée de détection~10m donc on peut mettre une caméra au 2e étage de l'atrium→angle assez large.

Inconvénients

-Il faut les mesures de l’atrium pour être sûr de ne pas sortir des limites (pas de limite par un tag au sol mais par une coordonnée limite)

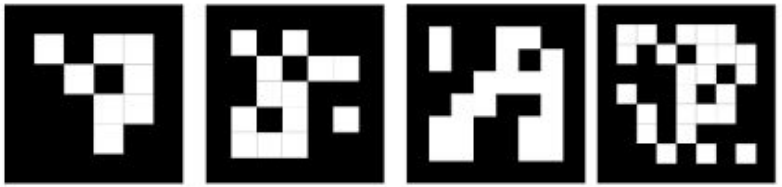
-Mesure en z beaucoup pas plus facile et peu précise

-Mesure à l'extérieur du dirigeable donc il faut envoyer les données pour pouvoir les traiter→cela peut ralentir le processus.

Utilisation de nouveaux tags: Aruco Markers

Raisons: -il fallait implémenter une nouvelle méthode: j'ai opté pour une bibliothèque plus simple à utiliser et au moins on a une comparaison possible

- Pas tant de différences de performances à part pour l'occlusion et les angles extrêmes (April Tags sont plus performants) → pas forcément un problème dans notre situation



Support de capture: Raspberry Pi 5 et PiCamera

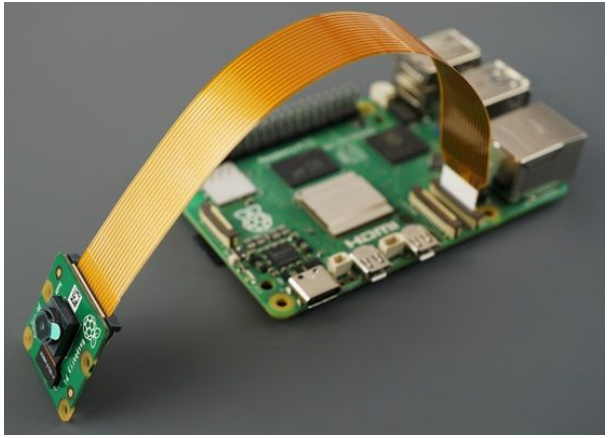
J’ai importé dans la raspberry un programme permettant:

-La prise continue et répétée de photos depuis la Pi Camera

-Sur chacune de ces photos déterminer la coordonnées en pixel dans le champ de caméra des tags 42 (dirigeable) et 1 (repère) grâce aux bibliothèques Aruco.

-L’obtention des coordonnées en pixel puis en millimètres dans ce repère

-L’obtention de l’inclinaison par rapport au repère



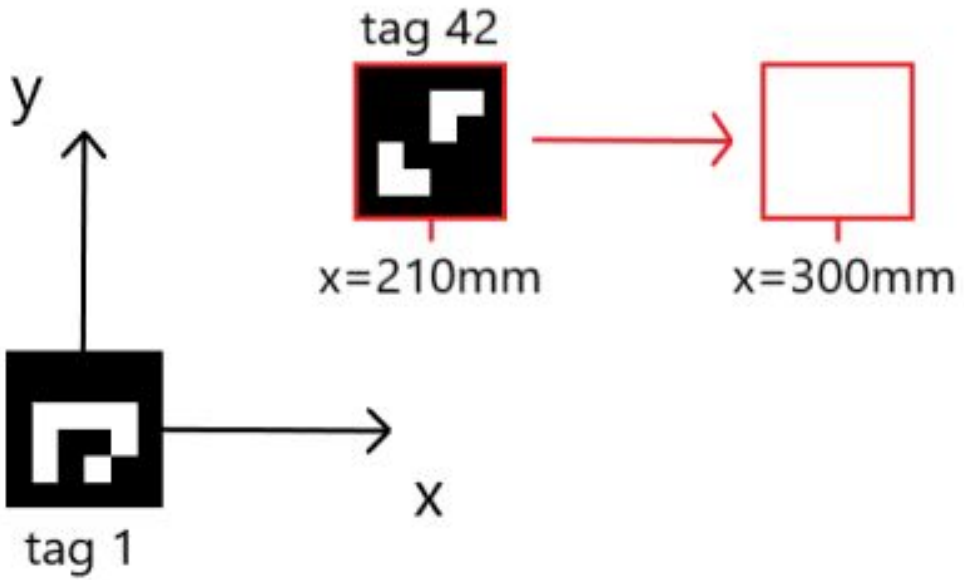
Mesures effectuées

Les mesures n'ont pas été effectuées sur le dirigeable directement.

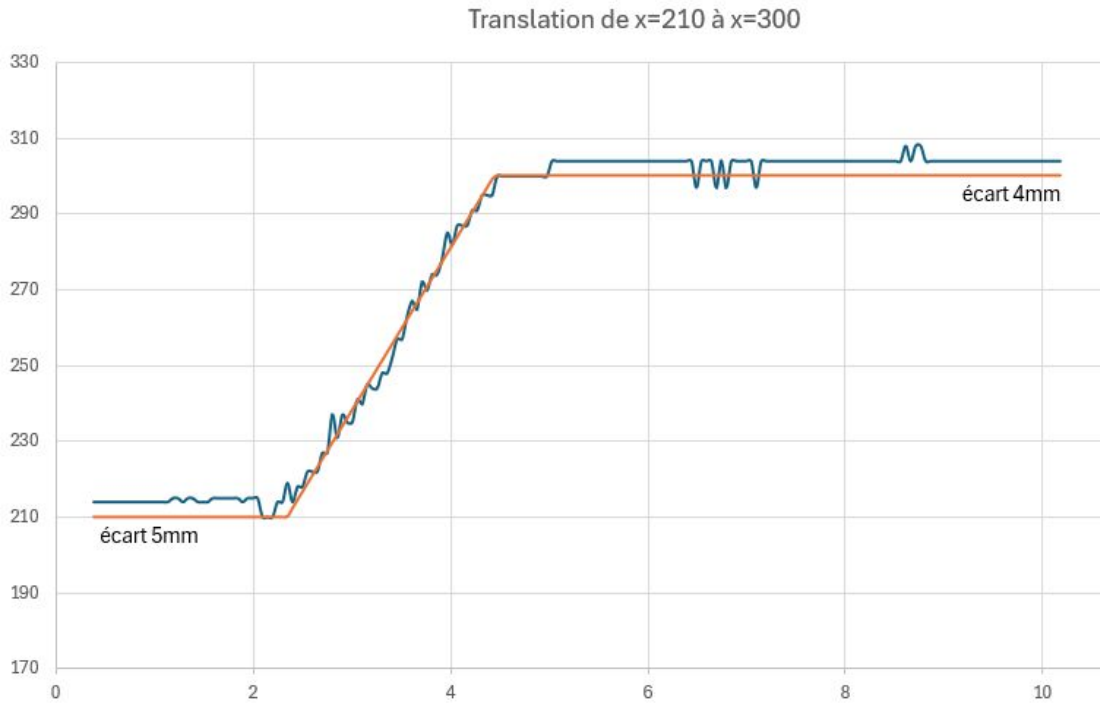
Parcours mesuré à l'avance: on déplace le tag 42 symbolisant le dirigeable sur ce parcours pour avoir une idée de la précision.

Observation: Mesure est moins précise quand le tag 42 s'éloigne du tag 1 et se rapproche des bords (quelques centimètres)

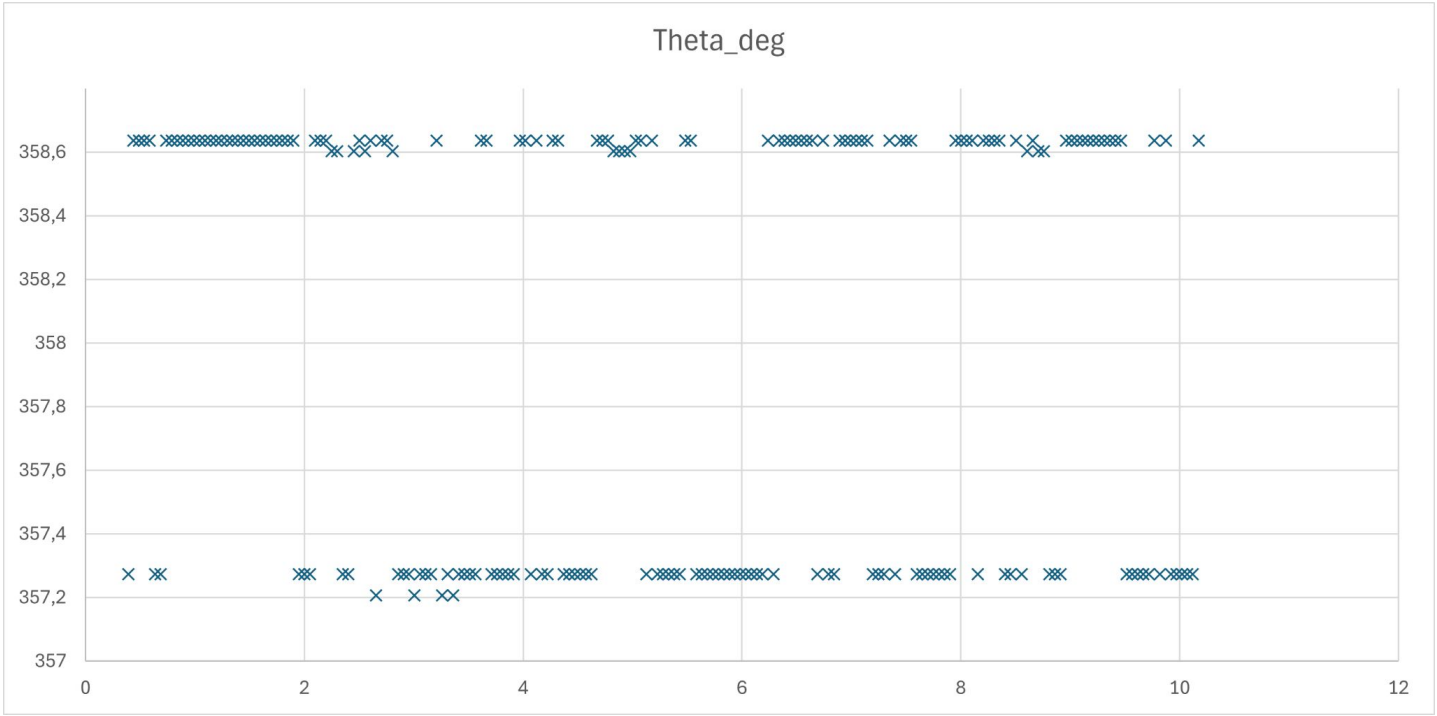
Exemple de parcours: translation horizontale



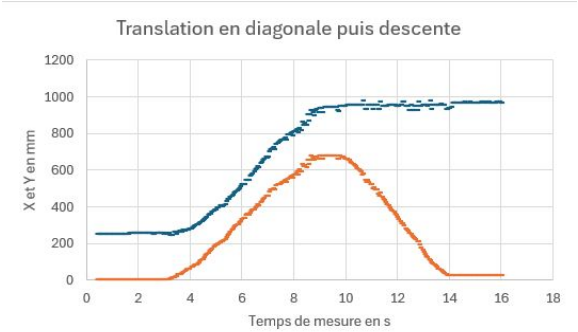
Résultats

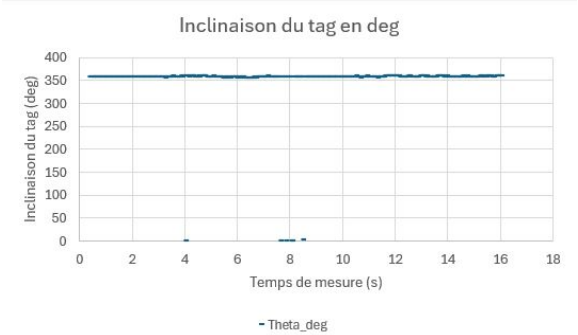

Translation de x=210 à x=300
330
310
290
270
250
230
210
190
170
0
2
4
6
8
10
écart 4mm
écart 5mm

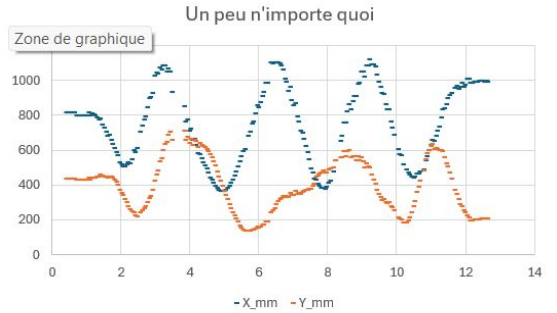
Résultats

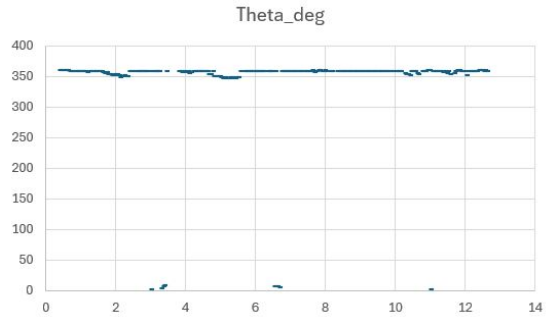

Theta_deg
358,6
358,4
358,2
358
357,8
357,6
357,4
357,2
357
0
2
4
6
8
10
12

Autres parcours mais pas de mesure précise donc pas d'estimation de l'erreur:

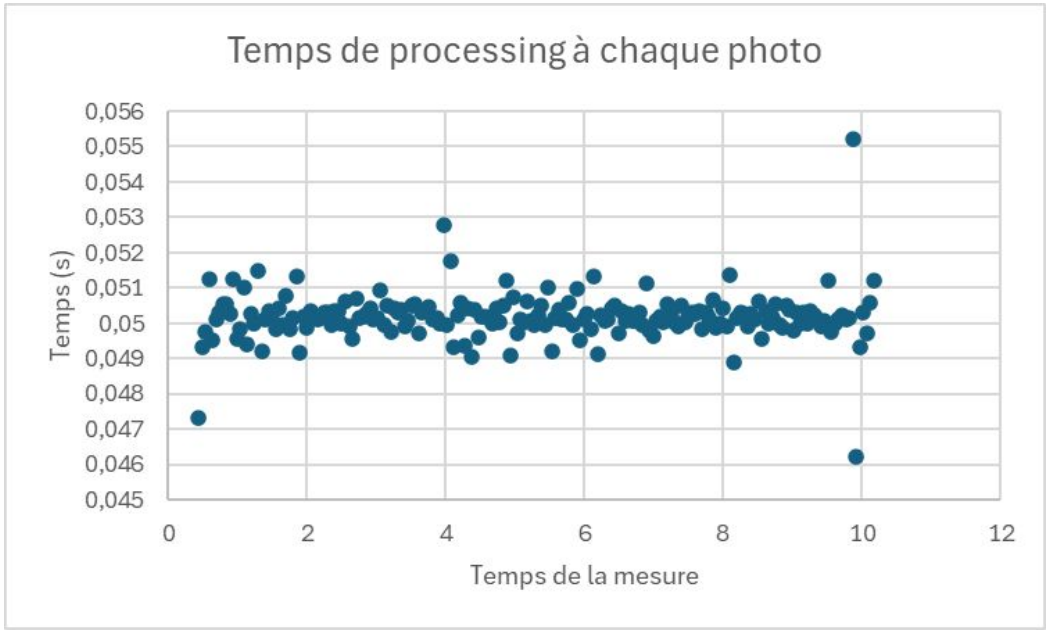








Temps de calcul



Conclusion

Plus pratique/esthétique à la mise en place: diminution du nombre de tag à placer au sol notamment.

Précision acceptable

Précision n'a pas été finie d'être testée et le transfère de données depuis la caméra externe non plus.

Solution des April Tags reste plus aboutie mais la nouvelle méthode pourrait apporter une nette amélioration.

GPS indoor

(MarvelMind)

Super-Beacon

4 Balise à ultrason fixe à placer sur une surface de 16 m2
1 Balise ultrason mobile

Protocole :

1) Installer Dashboard sur https://marvelmind.com/download/#SW ainsi que les driver nécessaire

2) Allumer 2=ON

3) Connecter beacon sur le pc en filaire

4) Upgrade le firmware de chaque balise et du modem en haut à gauche de l'écran

5) Cliquer sur le bouton default pour chaque beacon en bas à droite de l'écran

6) Définir une adresse différence pour chaque beacon (ex : 1,2,3,4,5) et cliquer sur write change en haut à droite de l'écran

7) Identifier chaque balise en mobile (Mobile Beacon a.k.a. "Hedgehog") ou fixe (Stationary Beacon) avec le hedgedog mode en enable ou disable .

8) Faire de même pour le modem (firmware+default button)

9) Placer les balises en mesurant les distances qui les sépares et leurs altitudes respectives.

10) Personnellement les balises sont placées à la même hauteur (0.75 m du sol)et forme une surface rectangulaire de 3,5x5,18 m^2 (voir carte associé)

11) Créer une submap

12) Réveiller chaque balise fixe dans la submap 0

13) Rentré manuellement leur position relative(x,y,z) car le mode placement auto n'est pas précis(clique droit sur le beacon sur la carte)

14) freezsubmap

15) Réveiller balise mobile et fixer son altitude

=> On peut maintenant se déplacer dans l’espace (voir vidéo test)

https://youtu.be/QFP-Thez-KA

Piste d’amélioration :

-placer les balises en hauteur orienté vers la zone à couvrir pour augmenter la zone et réduire les interférences du au passage des humains dans l’espace

-l’altitude ne marche pas (à régler)

-position peu précise(plusieurs frezze de position) (parfois la balise disparait)

=>peu fiable et robuste pour le moment

=> plus fiable/robuste avec 2 balise comme sur la vidéo tuto

-faire un essai/test sur dirigeable en mouvement

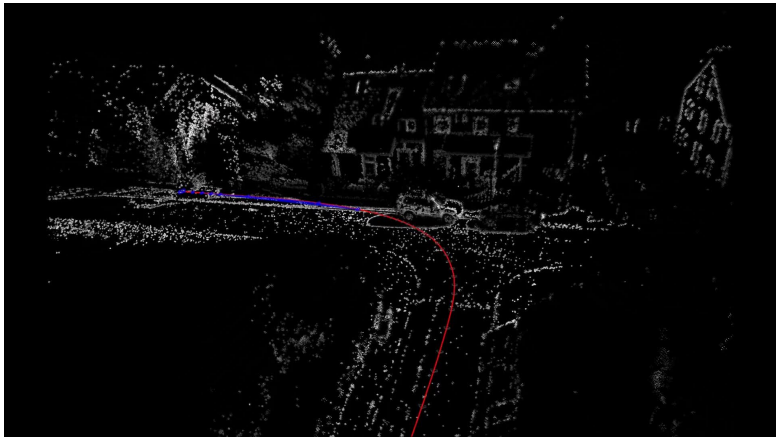
-intérêt pour le projet final ?

Vslam

(Visual Simultaneous Localization and Mapping)

Principe

La localisation et la cartographie simultanées visuelles (vSLAM) consistent à calculer **la position** et l'orientation d'une caméra par rapport à son **environnement** tout en **cartographiant** ce dernier. Ce processus utilise uniquement les entrées visuelles de la caméra.



Matériel

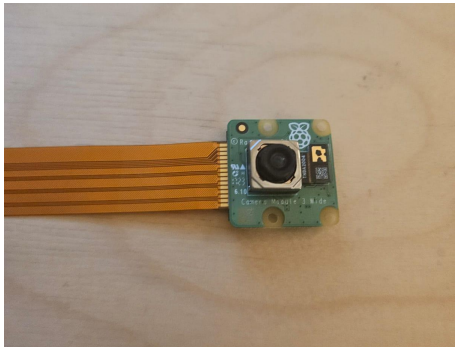
Raspberry Pi Camera :

monocular : utilise une seule caméra. Il capture des images 2D et se repose sur la structure acquise à partir du mouvement pour estimer la profondeur en comparant plusieurs images successives de la scène.

=> plus simple à mettre en place mais plus d'erreur

=> plus sensible à la lumière

=> nécessite un étalonnage précis



Matériel

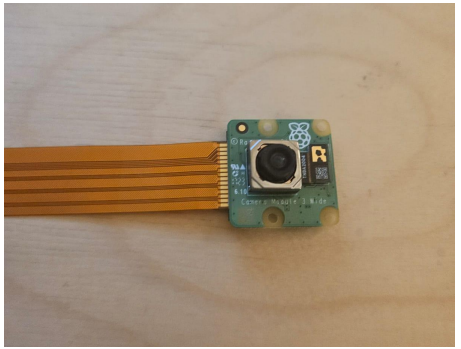
Raspberry Pi Camera :

monocular : utilise une seule caméra. Il capture des images 2D et se repose sur la structure acquise à partir du mouvement pour estimer la profondeur en comparant plusieurs images successives de la scène.

=> plus simple à mettre en place mais plus d'erreur

=> plus sensible à la lumière

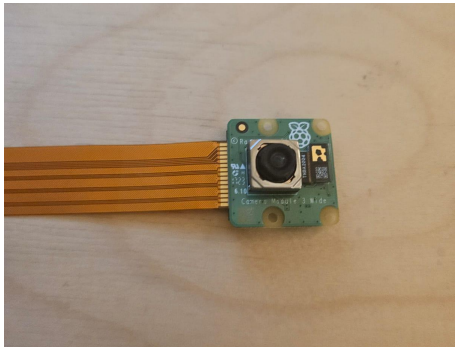
=> nécessite un étalonnage précis



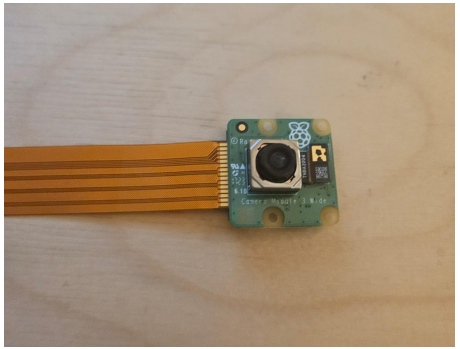
stero: Le VSLAM stéréo utilise deux caméras placées à une distance fixe l'une de l'autre, permettant une perception directe de la profondeur par triangulation à partir des images stéréo.

Matériel

Déterminer les caractéristiques :
-distances focales fx et fy
-centre de la caméra cx et cy

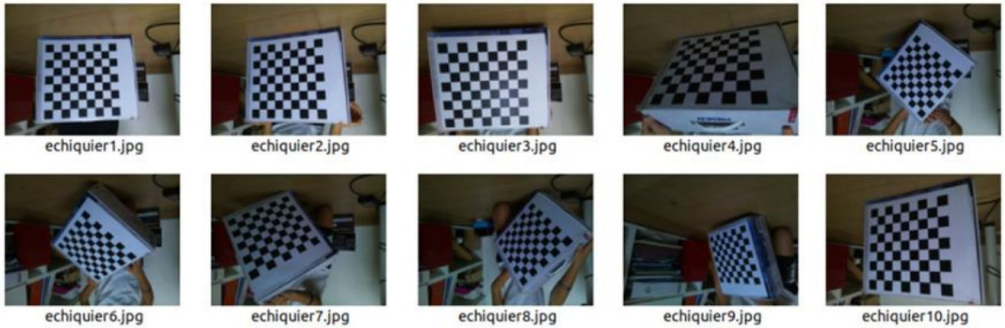


Matériel



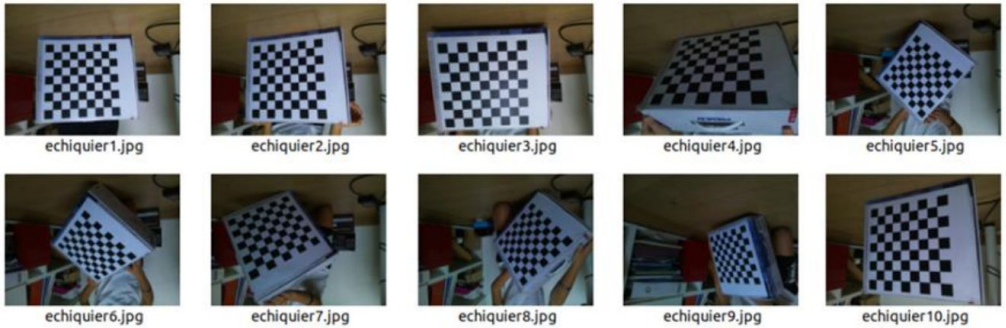
Déterminer les caractéristiques :
-distances focales fx et fy
-centre de la caméra cx et cy

=> utiliser le travail des prédecesseur pour déterminer la matrice intrinsèque à partir de 10 photo d'un échiquier



Elève de 4ème année au DER Nikola Telsa de l'ENS Paris-Saclay.

Matériel



Déterminer les caractéristiques :
-distances focales fx et fy
-centre de la caméra cx et cy

=> utiliser le travail des prédecesseur pour déterminer la matrice intrinsèque à partir de 10 photo d'un échiquier

Matrice intrinseque :

[[1.93955859e+03 0.00000000e+00 2.33028002e+03]

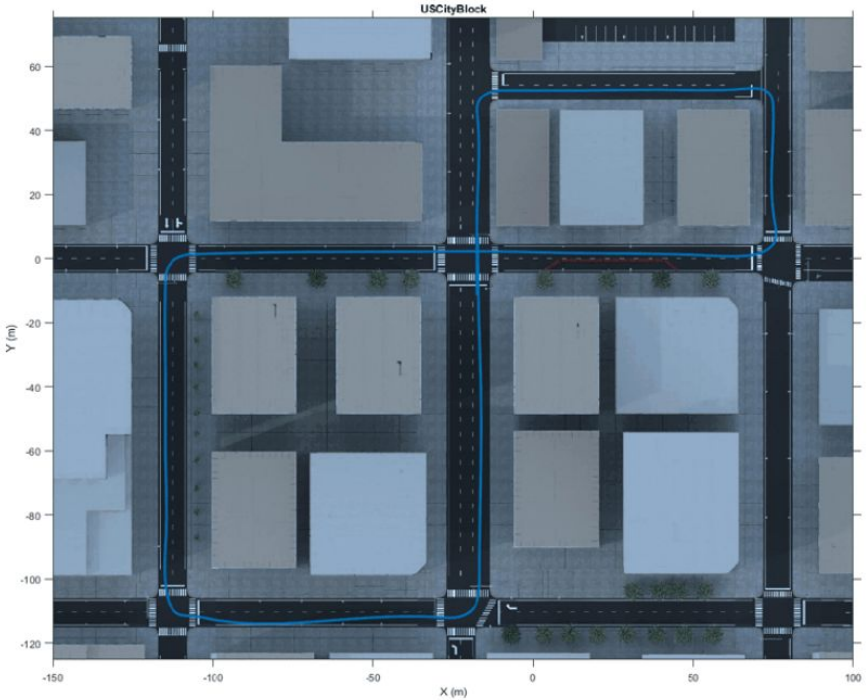
[0.00000000e+00 1.94784289e+03 1.31728307e+03]

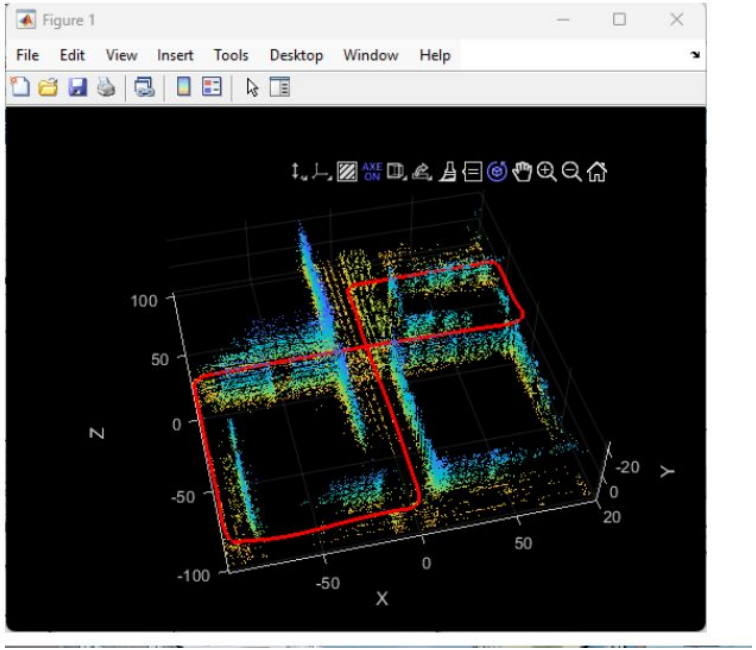
[0.00000000e+00 0.00000000e+00 1.00000000e+00]]

Coefficients de distorsion :

[[-0.08220333 0.15349617 0.00434731 0.00802244
-0.11979576]]

Matlab-Orb-slam





Matlab-Orb-slam

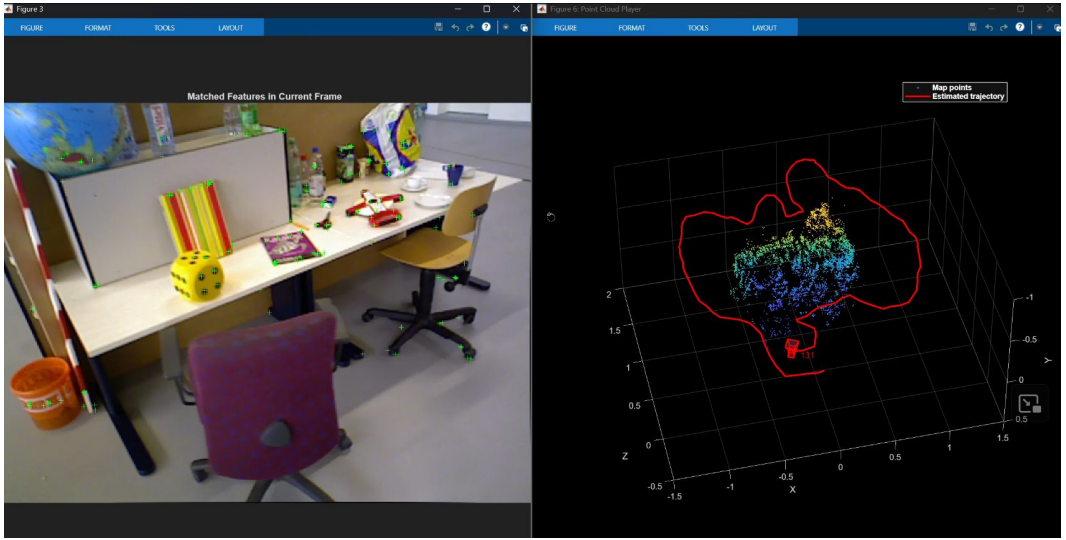
Étape pour construire un pipeline de SLAM visuel basé sur les points caractéristiques à partir d'une séquence d'images :

- Initialiser la carte — Initialiser la carte 3D des points à partir de deux images.
- Suivre les caractéristiques — Pour chaque nouvelle image, on estimer la pose de la caméra en associant les caractéristiques de l'image courante avec celles de la dernière image clé.
- Créer une carte locale — on créer une nouvelle carte 3D de points.
- Détecter les boucles — Détecter les boucles en comparant l'image clé courante avec toutes les images clés précédentes pour se repérer sur la carte(via une approche basée sur bag-of-features).

Matlab-Orb-slam

=> Application sur un benschmark

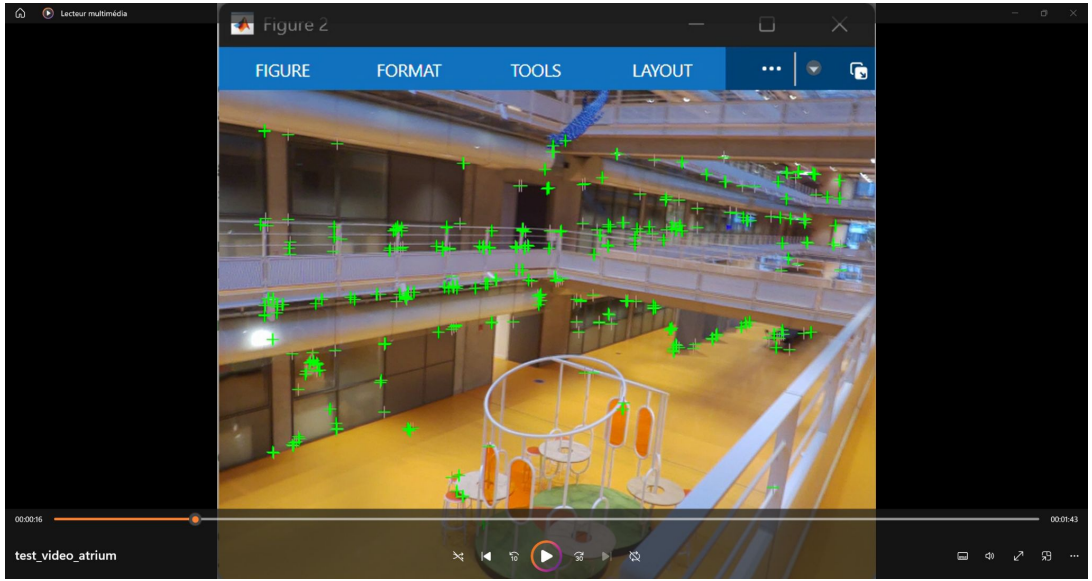
https://youtu.be/ShuiijPZmd8



Matlab-Orb-slam

=> Application à l'atrium

https://youtu.be/KTA8TM7LTpc



conclusion **Visual SLAM**

Le Vslam permettraient effectivement d'obtenir une cartographie de l'atrium et de se localiser dans ce dernier à l'aide d'une simple caméra.

Piste d’amélioration:

=>peaufiner le modèle matlab pour obtenir la carte de l’atrium

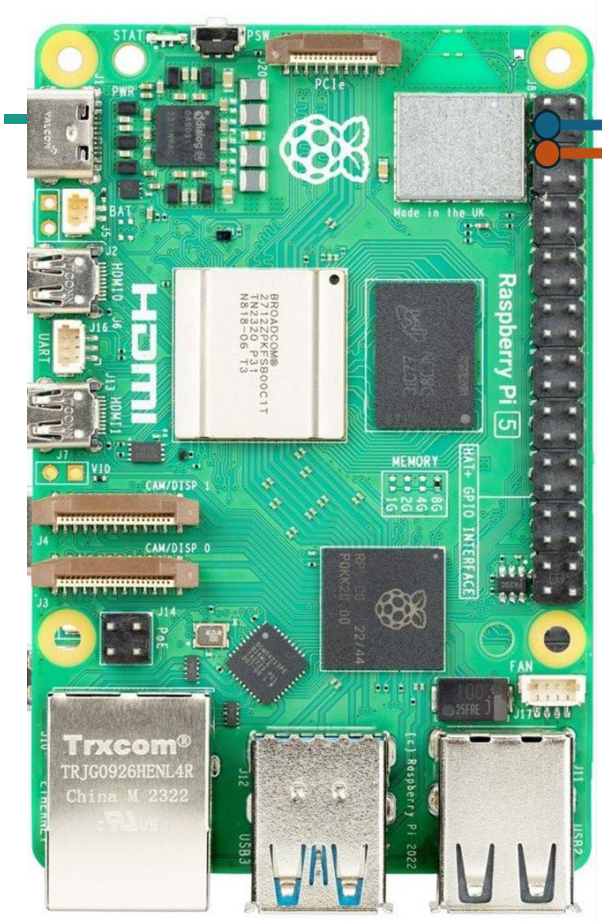
=> utiliser une caméra stéréo

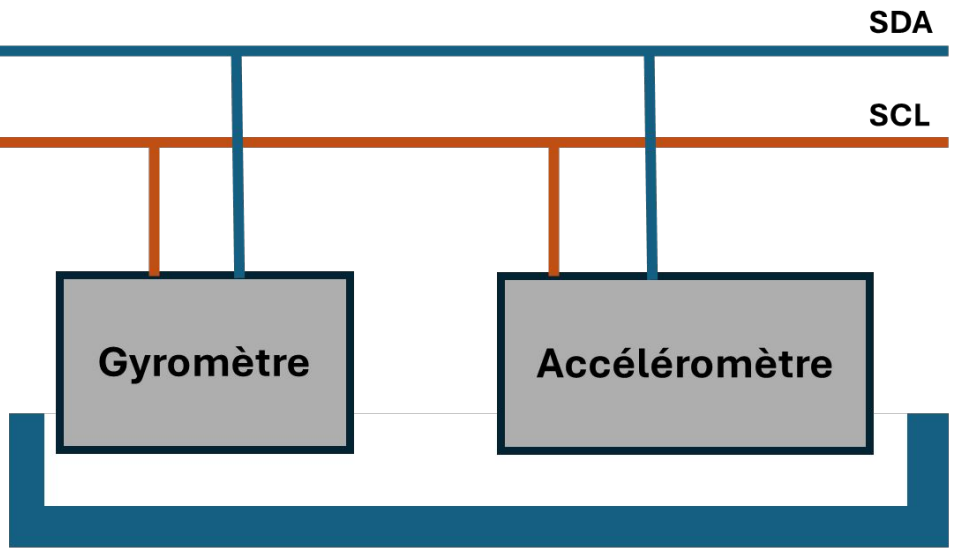
=> envisager le Slam classique (ex:via lidar)

Centrale inertielle

Critère	BNO055(2024-2025)	BMI088 (2025-2026)
précision accéléromètre	1 mg	0.09 mg
précision gyroscope	16°/s	0.004°/s
Fusion des données intégrée	OUI	NON
Bruit accéléromètre	moyen	faible
Bruit gyroscope	très faible	faible
Disponibilité de bibliothèques	OUI	NON

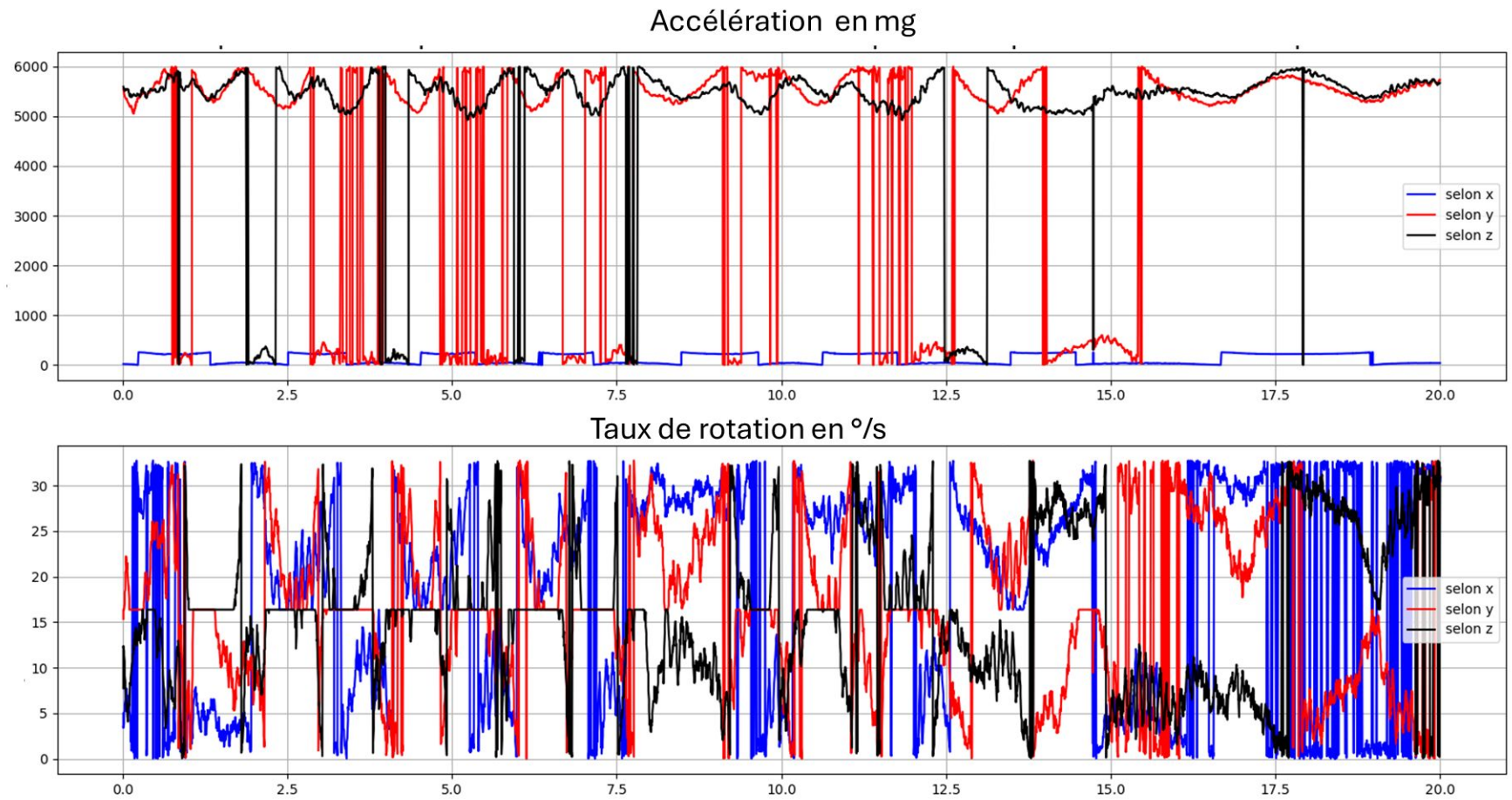
Centrale inertielle



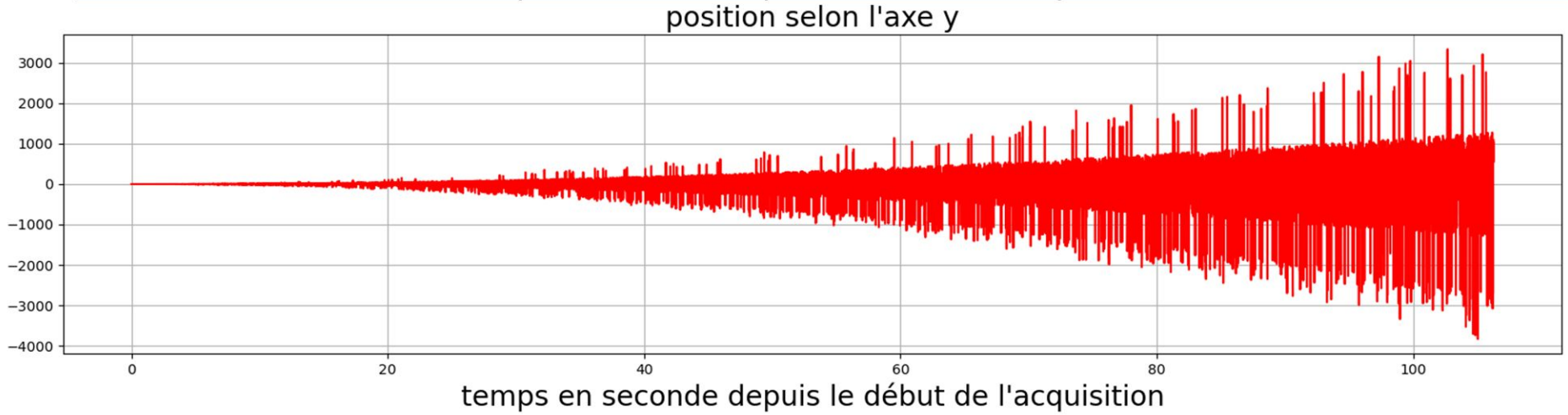


BMI088

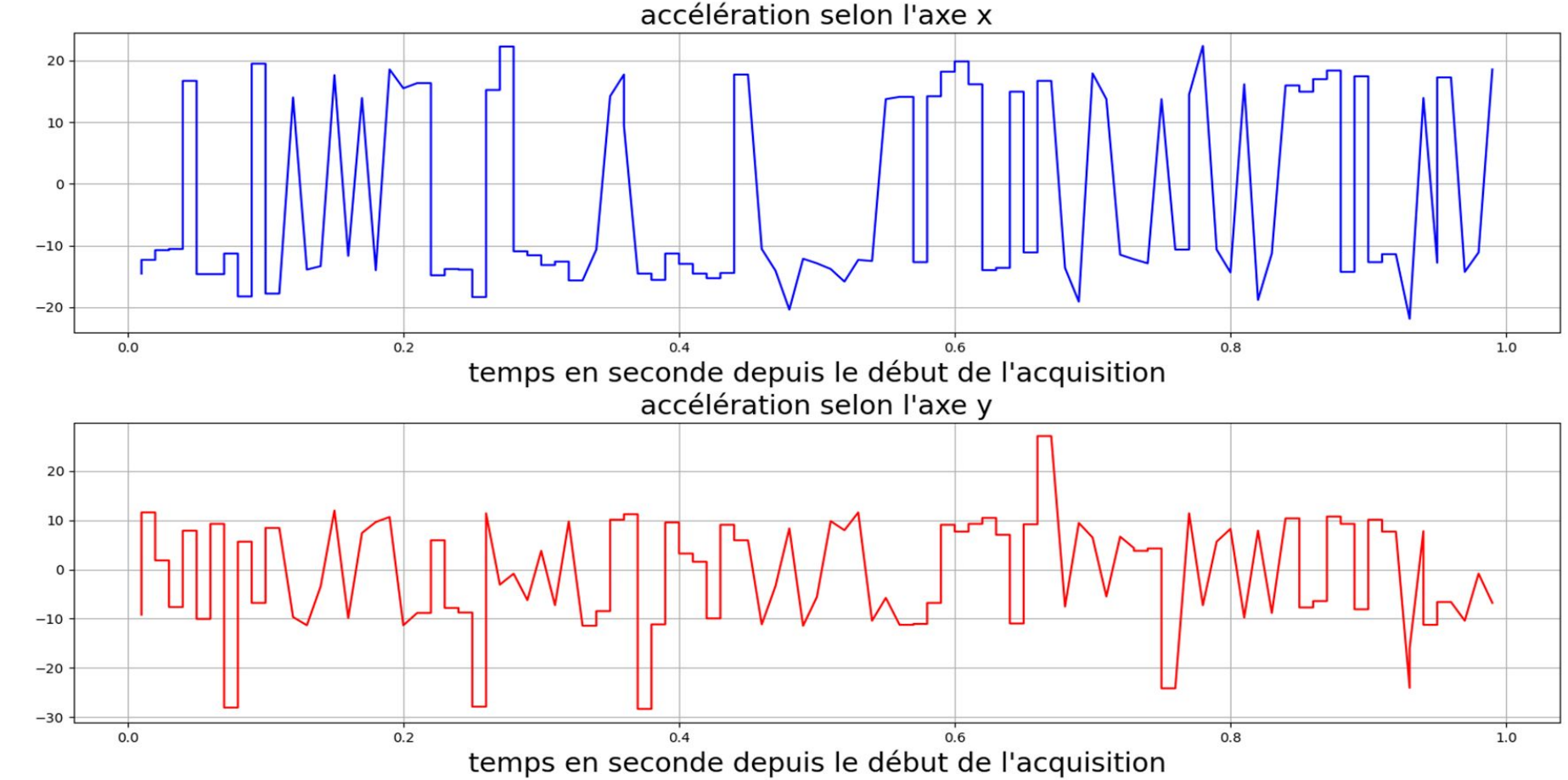
Centrale inertielle- premiers essais



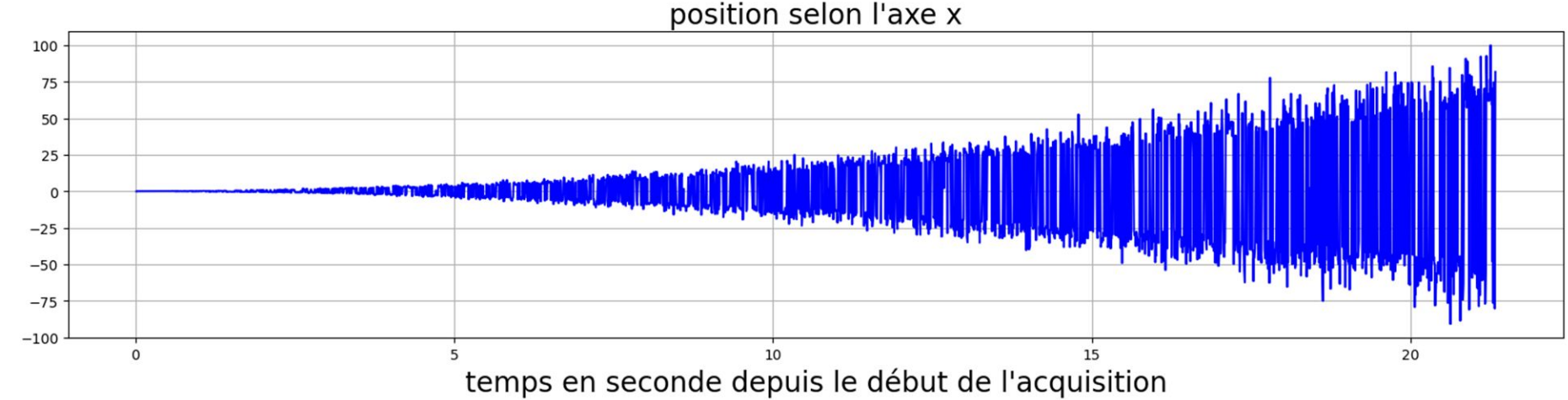
Centrale inertielle- premiers essais

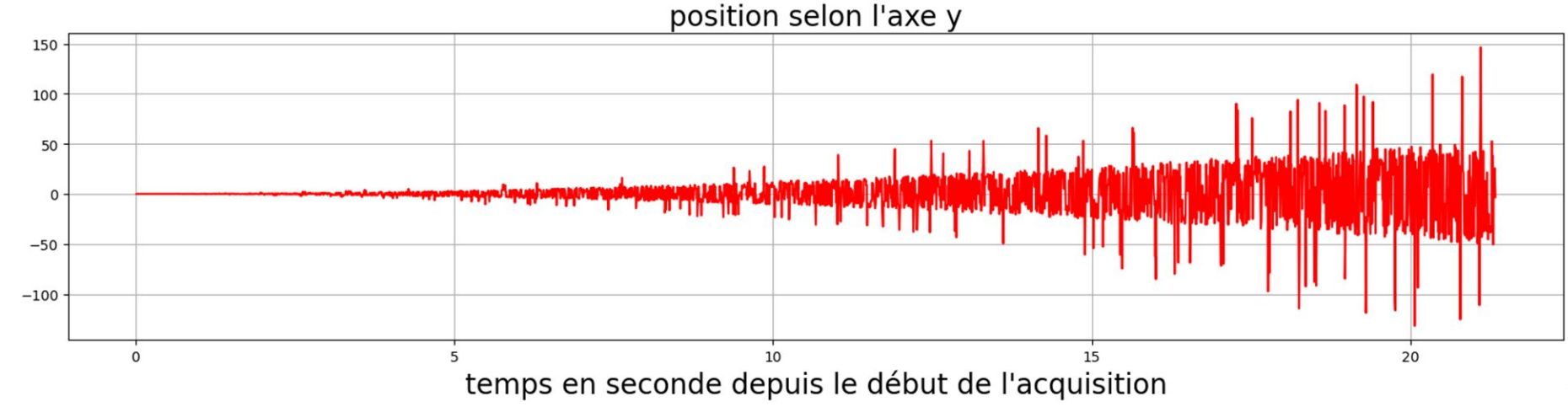


Centrale inertielle- essais statiques

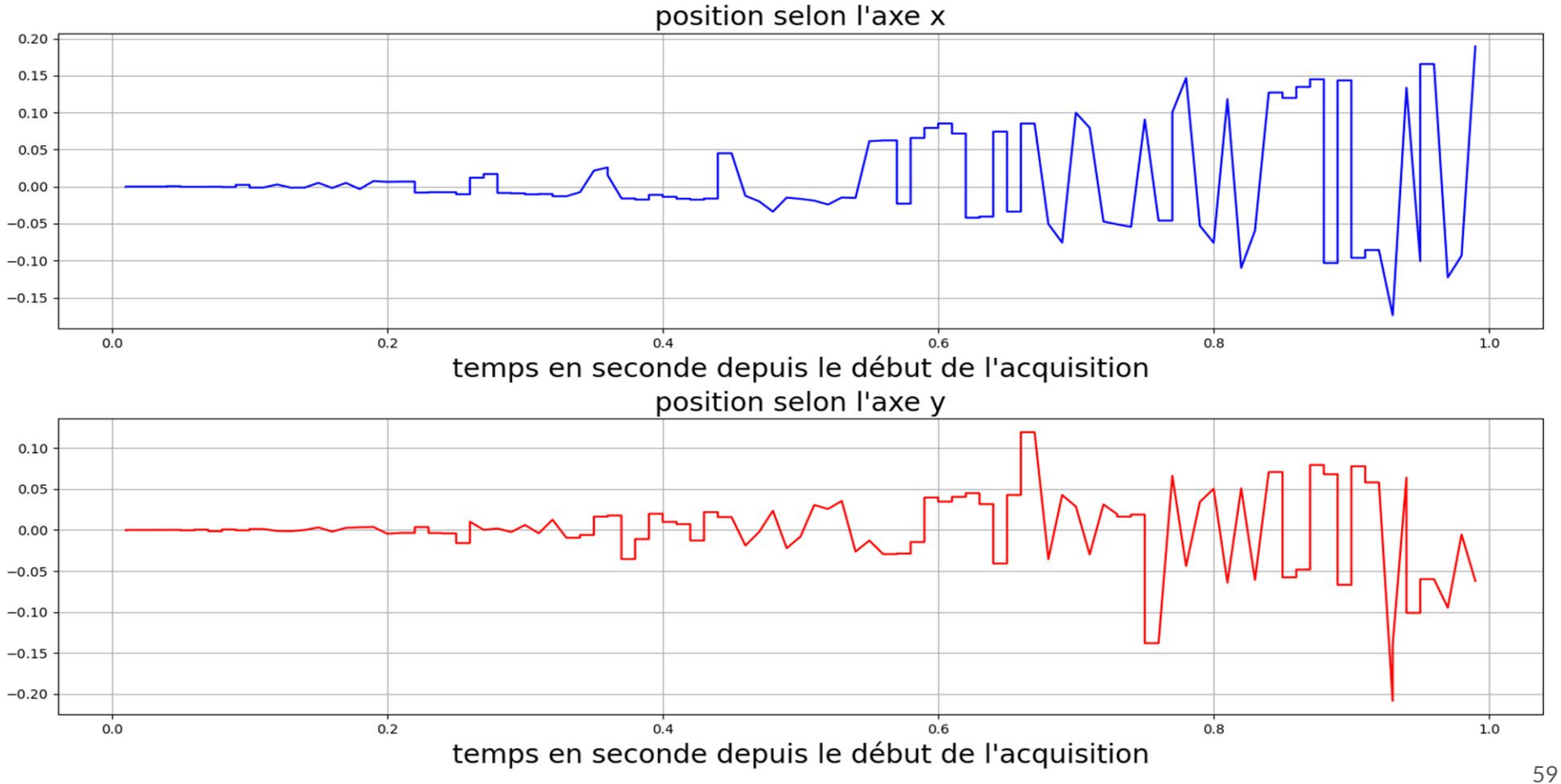


Centrale inertielle- essais statiques





Centrale inertielle- essais statiques



Pistes d'amélioration des performances

- Mise en place d'une phase de calibration permettant d'évaluer et de compenser l'offset
- (Re)filtrage de sorties par un passe-bas
- Fusion des données du capteur

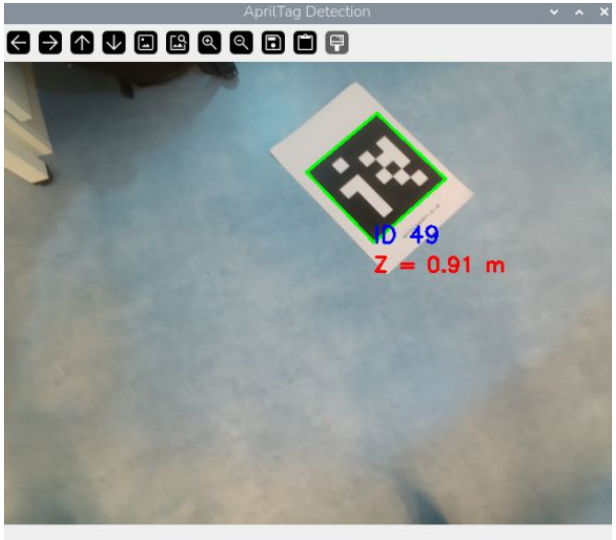
Localisation par la méthode des Apriltags au sol

Étude de ce qui a été fait l'année dernière pour essayer d'améliorer la précision de la détection.

- Prise de photo d'une caméra
- Calibration d'une caméra
- Détection d'un Apriltag
- Comparaison des données mesurées aux données réelles

Localisation par la méthode des Apriltags au sol

Programme de détection



Localisation par la méthode des Apriltags au sol

Programme de détection

- Mode preview -> Mode vidéo
- Précision sur la famille des Apriltags utilisée
- Précision sur les 4 coeurs disponibles sur la RPI
- Précision sur d'autres paramètres…

```
43  detector = Detector(
44      families="tag36h11",
45      nthreads=4,           # Pi 5 = 4 coeurs
46      quad_decimate=2.0,    # gain de vitesse
47      quad_sigma=0.0,
48      refine_edges=True, # plus rapide
49      decode_sharpening=0.25
50  )
```

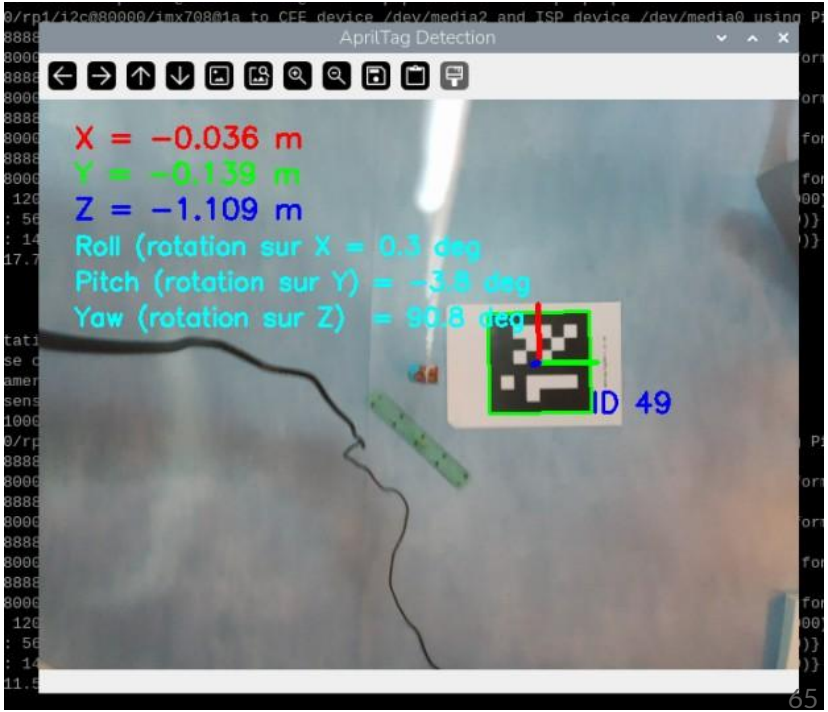
```
Tags détectés : 1 | Temps acquisition : 629.63 ms | Temps détection : 13.50 ms | Temps total : 643.13 ms
Tags détectés : 1 | Temps acquisition : 60.83 ms | Temps détection : 12.29 ms | Temps total : 73.12 ms
Tags détectés : 1 | Temps acquisition : 48.99 ms | Temps détection : 9.95 ms | Temps total : 58.94 ms
Tags détectés : 1 | Temps acquisition : 53.76 ms | Temps détection : 7.64 ms | Temps total : 61.40 ms
Tags détectés : 1 | Temps acquisition : 59.81 ms | Temps détection : 7.42 ms | Temps total : 67.23 ms
Tags détectés : 1 | Temps acquisition : 61.27 ms | Temps détection : 10.87 ms | Temps total : 72.15 ms
Tags détectés : 1 | Temps acquisition : 56.49 ms | Temps détection : 17.30 ms | Temps total : 73.79 ms
Tags détectés : 1 | Temps acquisition : 53.12 ms | Temps détection : 12.49 ms | Temps total : 65.61 ms
Tags détectés : 1 | Temps acquisition : 56.20 ms | Temps détection : 11.77 ms | Temps total : 67.97 ms
Tags détectés : 1 | Temps acquisition : 57.99 ms | Temps détection : 11.90 ms | Temps total : 69.89 ms
Tags détectés : 1 | Temps acquisition : 57.47 ms | Temps détection : 11.90 ms | Temps total : 69.37 ms
Tags détectés : 1 | Temps acquisition : 57.52 ms | Temps détection : 11.79 ms | Temps total : 69.32 ms
Tags détectés : 1 | Temps acquisition : 57.99 ms | Temps détection : 11.70 ms | Temps total : 69.70 ms
Tags détectés : 1 | Temps acquisition : 58.04 ms | Temps détection : 7.37 ms | Temps total : 65.41 ms
```

```python
while True:
    start = time.time()

    frame = picam2.capture_array()
    t1 = time.time() - start
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray_undistorded = cv2.undistort(gray, mtx, dist, None, newCameraMatrix=mtx)
    tags = detector.detect(gray_undistorded)
    t2 = time.time() - start - t1
    t3 = time.time() - start
    print(f"Tags détectés : {len(tags)} | Temps acquisition : {t1*1000:.2f} ms | Temps détection : {t2*1000:.2f} ms | Temps total : {t3*1000:.2f} ms")


    if cv2.waitKey(1) == 27:
        break
```

Localisation par la méthode des Apriltags au sol

Localisation dans l'espace

- Affichage des données de positions et d'inclinaisons de l'apriltag par rapport à la caméra
- Offset bien réglé
- Mesure de la position réelle grâce à un support et une masselotte suspendue à la caméra
- Problème dès qu'on s'éloigne de l'origine
- Problème dès qu'il y a un léger obstacle entre la caméra et l'apriltag



Localisation par la méthode des Apriltags au sol

Pistes d'améliorations

- Utiliser le damier de l'ENS pour la calibration
- Équiper ma caméra d'une batterie pour plus de liberté
- Changer de caméra
- Travailler directement avec la caméra qui est sur cobrasalma pour tests et améliorations de la précision