Visualisation des gestes techniques en patinage

Comment visualiser ses gestes en patinage pour les améliorer ?

Le patinage

- Des jouets pour apprendre l'équilibre aux enfants ou se promener en ville ou en forêt

- Des fédérations organisant des compétitions jusqu'au niveau international

Fédération Française Roller & Skateboard

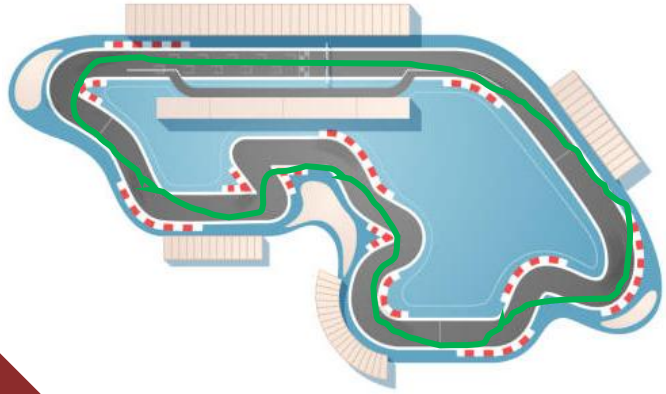
SPEED SKATING WORLD CHAMPIONSHIPS COLOMBIA 2021

Sport in the Olympic programme

Crédits
Roller : site web Decathlon.fr
Logos : site web Olympics.com

Améliorer sa vitesse en patinage

- Avoir une gestuelle parfaite
- Conserver une position idéale sur la piste
- Tenir une cadence



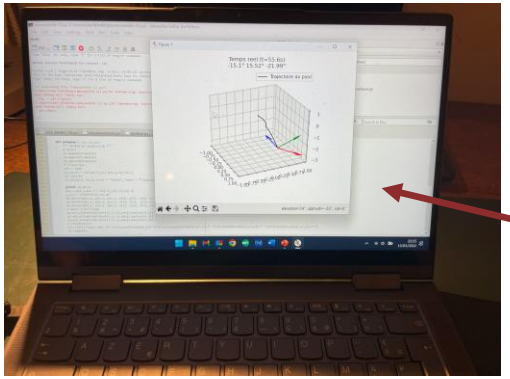
Crédits
Circuit : iStock
Photo : site web FFR

Comment visualiser ses gestes en patinage ?

- Visualiser en temps réel (pour corriger)
- Visualiser a posteriori (pour comparer)

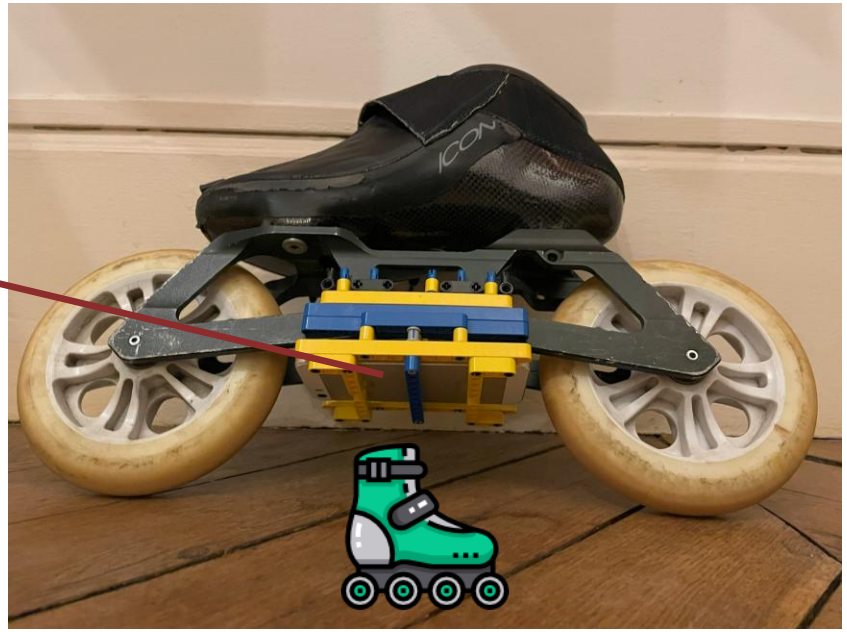
- Recueillir des données (temporels et spatiales 3D)
- Capteurs physiques (≠ caméras)

Choix du système expérimental et budget



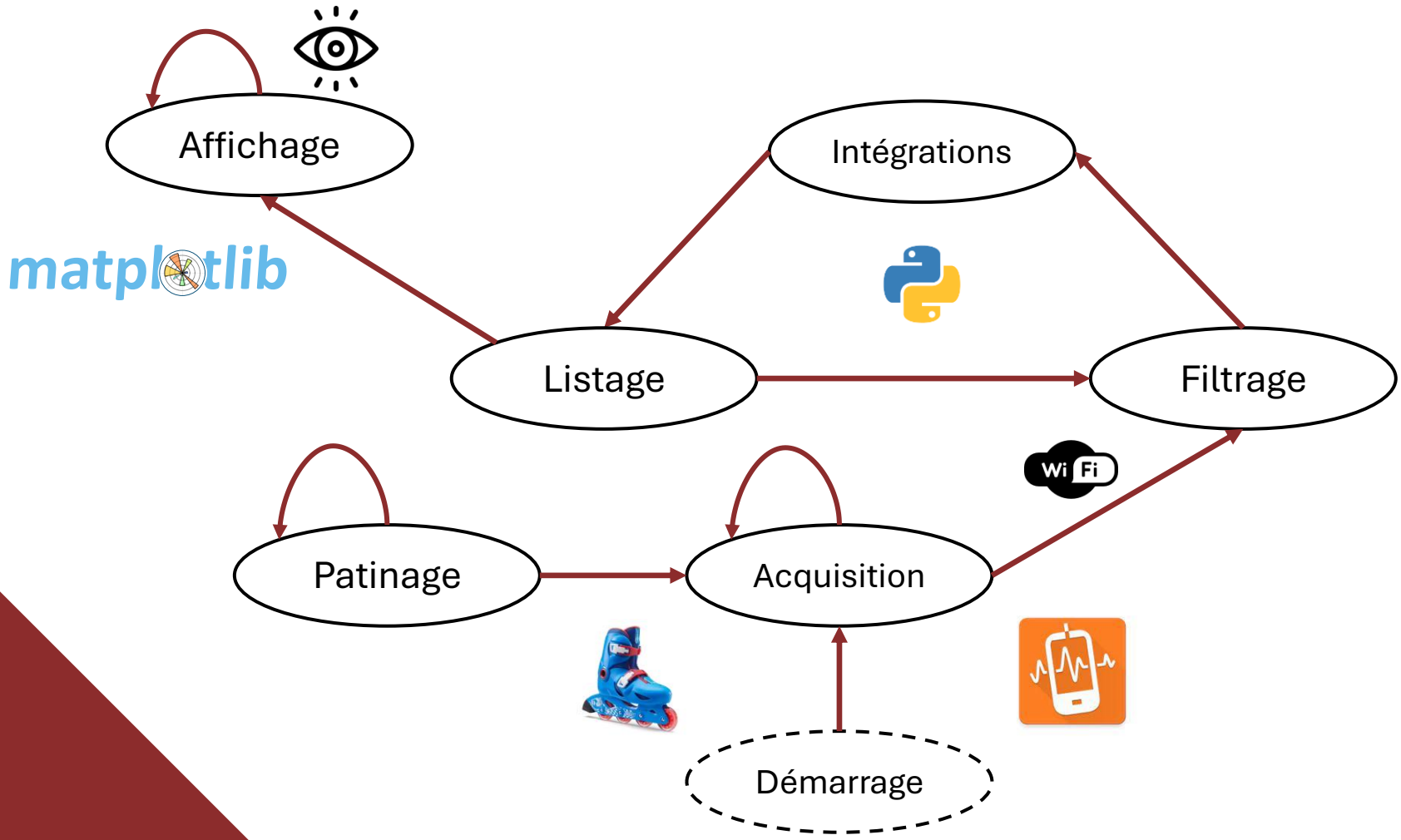
PC de traitement
et de visualisation

Liaison
sans fils



Roller avec le smartphone-capteur fixé

Fonctionnement expérimental



Crédits
Pictogrammes : iStock/AppStore
Roller : site web Decathlon.fr

Récupération des données

- Démarrage de l'acquisition

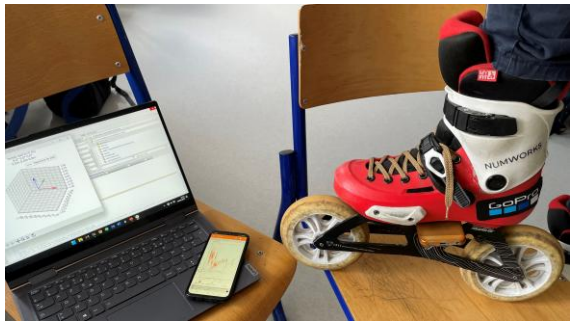
```
for action in ["stop","clear","start"]:
    r.get(url="http://"+ip+"/control?cmd="+action)
```

- Récupération des données en continue

```
while True:
    tpg,axpg,aypg,azpg,wxpg,wypg,wzpg = requete(ip,["acc_time","accX","accY","accZ","gyrX","gyrY","gyrZ"])
```

Appel de l'API Phyphox

```
def requete(IPPORT,demandephyphox):
    """ requete phyphox """
    # Creation de la requete
    demande = ""
    for capteur in demandephyphox:
        demande+=capteur+"&"
    requete = "http://"+IPPORT+"/get?"+demande
    # Envoie de la requete
    try :
        data = r.get(url=requete).json()
    except Exception as erreur:
        pass
    # Reception de la requete
    reponse = []
    for capteur in demandephyphox:
        try :
            reponse += [data["buffer"][capteur]["buffer"][0]]
        except Exception as erreur:
            pass
    if reponse[-1]==None:
        return [0.,0.,0.,0.,0.,0.,0.]
    return reponse
```



Filtrage des données

$$Filtre\ passe-bande\ de\ l'accélération\ suivant\ x : \begin{cases} \lim\limits_{a_x \to +\infty} |a_x| = 0 \\ \lim\limits_{a_x \to 0} |a_x| = 0 \end{cases}$$

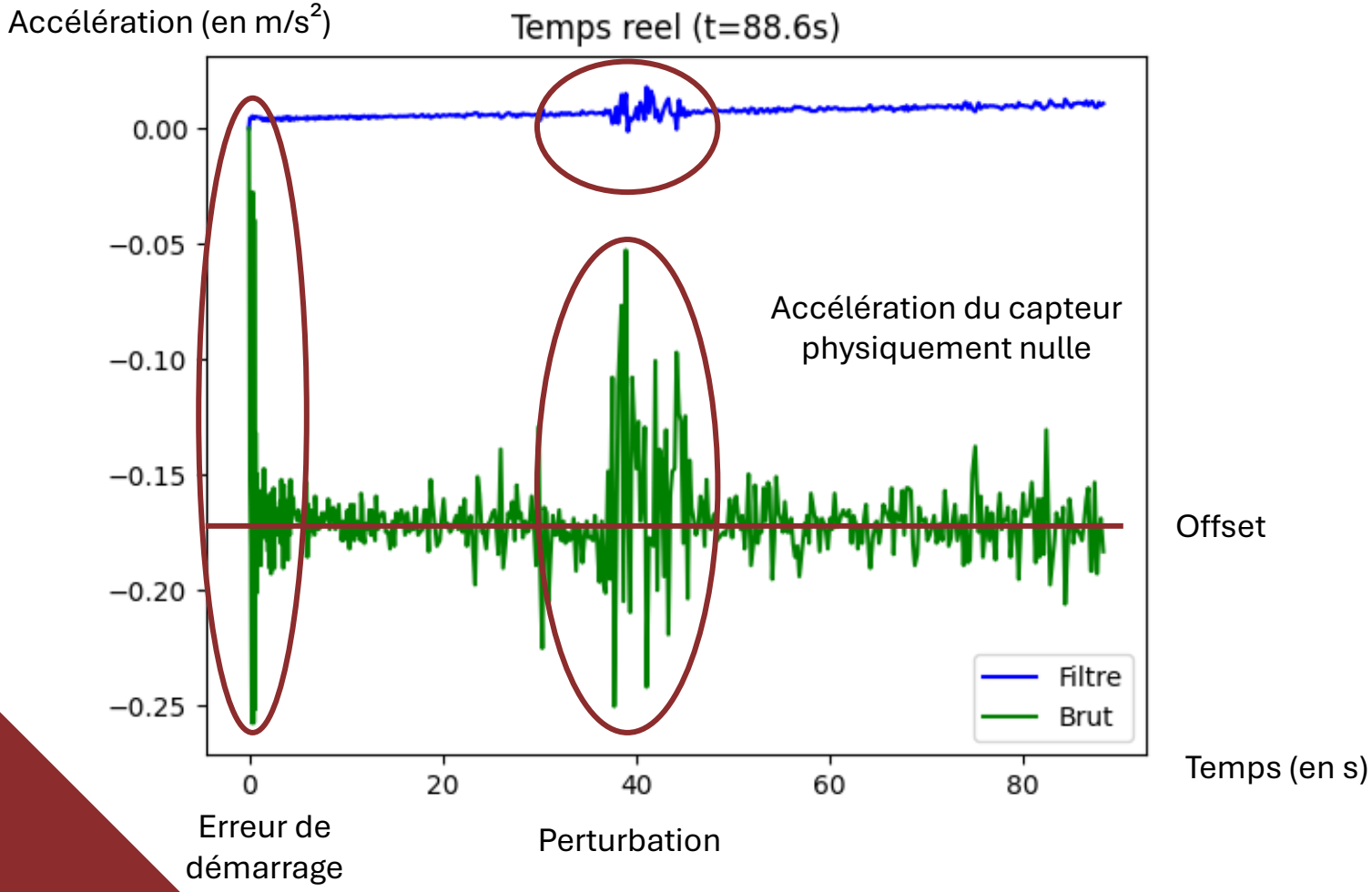
```python
def abbe(valeur,max):
    if abs(valeur) >= max:
        return max
    return valeur
```

```python
def epsilon(valeur,n):
    """ "0" """
    if abs(valeur) <= 10**(-n):
        return 0.
    return valeur
```

```python
def moy_gliss(d,L):
    """ filtre de la moyene glissante """
    l = len(L)
    if l == n:
        for i in range(1,n-1):
            L[i+1]=L[i]
        L[1]= d
        return sum(L)/l
    else:
        L.append(d)
        return sum(L)/(l+1)
```

```python
def calibrage(L):
    """ Moyenne """
    A = [sum([L[i][k] for i in range(len(L))])/len(L) for k in range(3)]
    return A[0],A[1],A[2]
```

Graphe du filtrage des données



Equations horaires tridimensionnelles et intégrations

$$Position\ angulaire\ autour\ de\ x : \theta_x(t_n) = \sum_{0}^{n-1} \theta_x(t) + \int_{n-1}^{n} \omega_x(t) \cdot dt$$

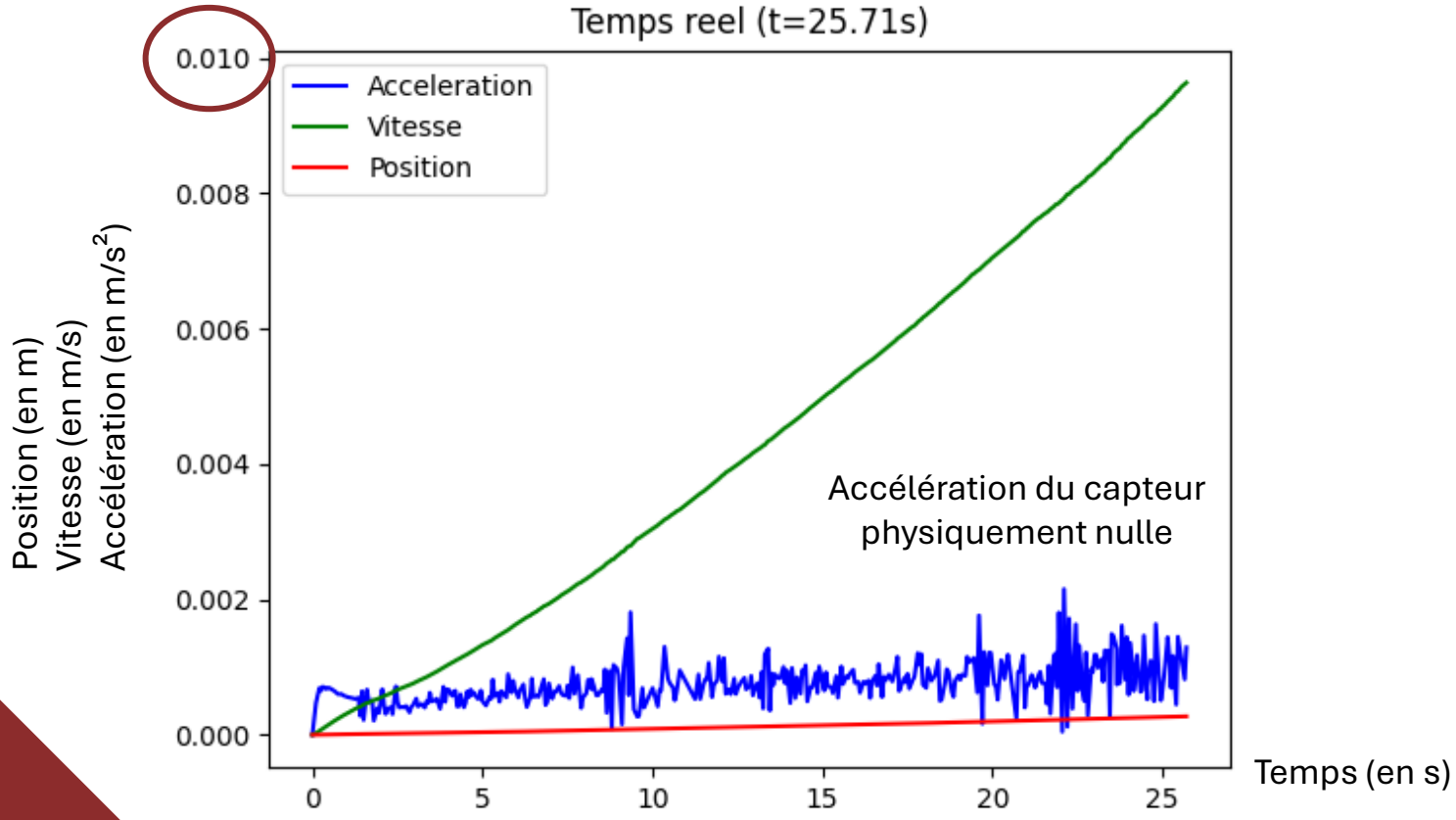
```
tpg,axpg,aypg,azpg,wxpg,wypg,wzpg = requete(ip,["acc_time","accX","accY","accZ","gyrX","gyrY","gyrZ"])
Opg += np.trapz([tampon[-3],wxpg],x=[tampon[0],tpg]) # angle autour de x
Fpg += np.trapz([tampon[-2],wypg],x=[tampon[0],tpg]) # angle autour de y
Ppg += np.trapz([tampon[-1],wzpg],x=[tampon[0],tpg]) # angle autour de z
```

$$Position\ suivant\ x : p_x(t_n) = \sum_{0}^{n-1} p_x(t) + \iint_{n-1}^{n} a_x(t) \cdot dt^2$$

```
vxpg += np.trapz([tampon[1],axpg],x=[tampon[0],tpg])
vypg += np.trapz([tampon[1],aypg],x=[tampon[0],tpg])
vzpg += np.trapz([tampon[1],azpg],x=[tampon[0],tpg])

   x += np.trapz([xpg[-1],xxpg[0]+yypg[0]+zzpg[0]],x=[tampon[0],tpg])
   y += np.trapz([ypg[-1],xxpg[1]+yypg[1]+zzpg[1]],x=[tampon[0],tpg])
   z += np.trapz([zpg[-1],xxpg[2]+yypg[2]+zzpg[2]],x=[tampon[0],tpg])
```

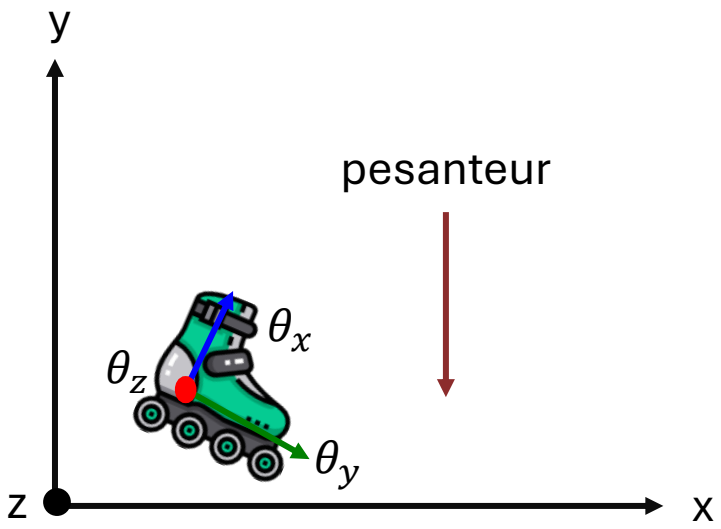
Graphes des équations horaires et intégrations



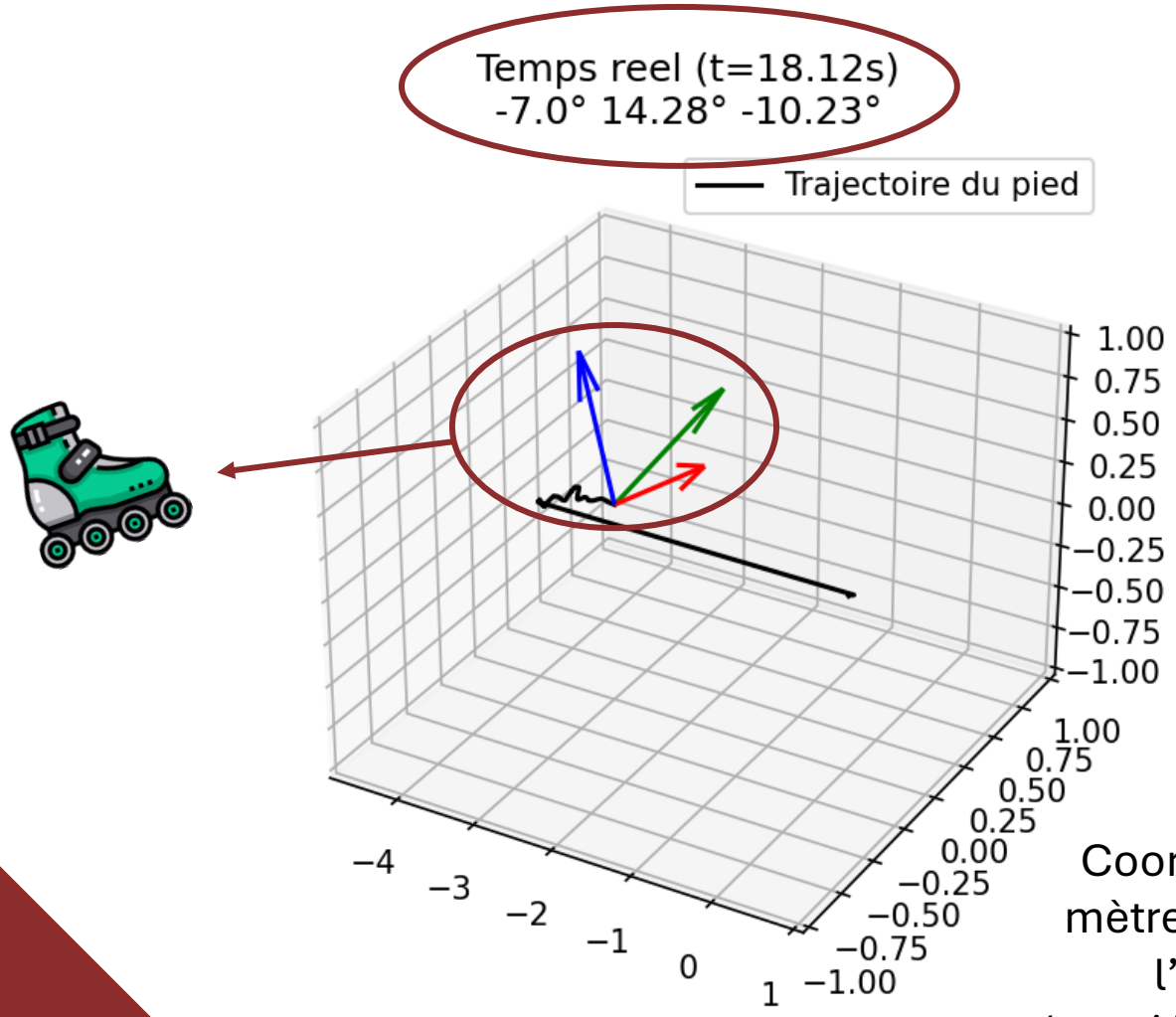
Problématiques d'affichage

$$\textit{Changement de base : sphériques à cartésiennes} : \begin{cases} x = r \cdot \sin\theta_x \cdot \cos\theta_z \\ y = r \cdot \sin\theta_x \cdot \sin\theta_z \\ z = r \cdot \cos\theta_x \end{cases}$$

```python
def spheriques_cartesiennes(norme,angle_autour_z,angle_autour_x):
    x = norme*np.sin(angle_autour_x)*np.cos(angle_autour_z)
    y = norme*np.sin(angle_autour_x)*np.sin(angle_autour_z)
    z = norme*np.cos(angle_autour_x)
    return x,y,z
```



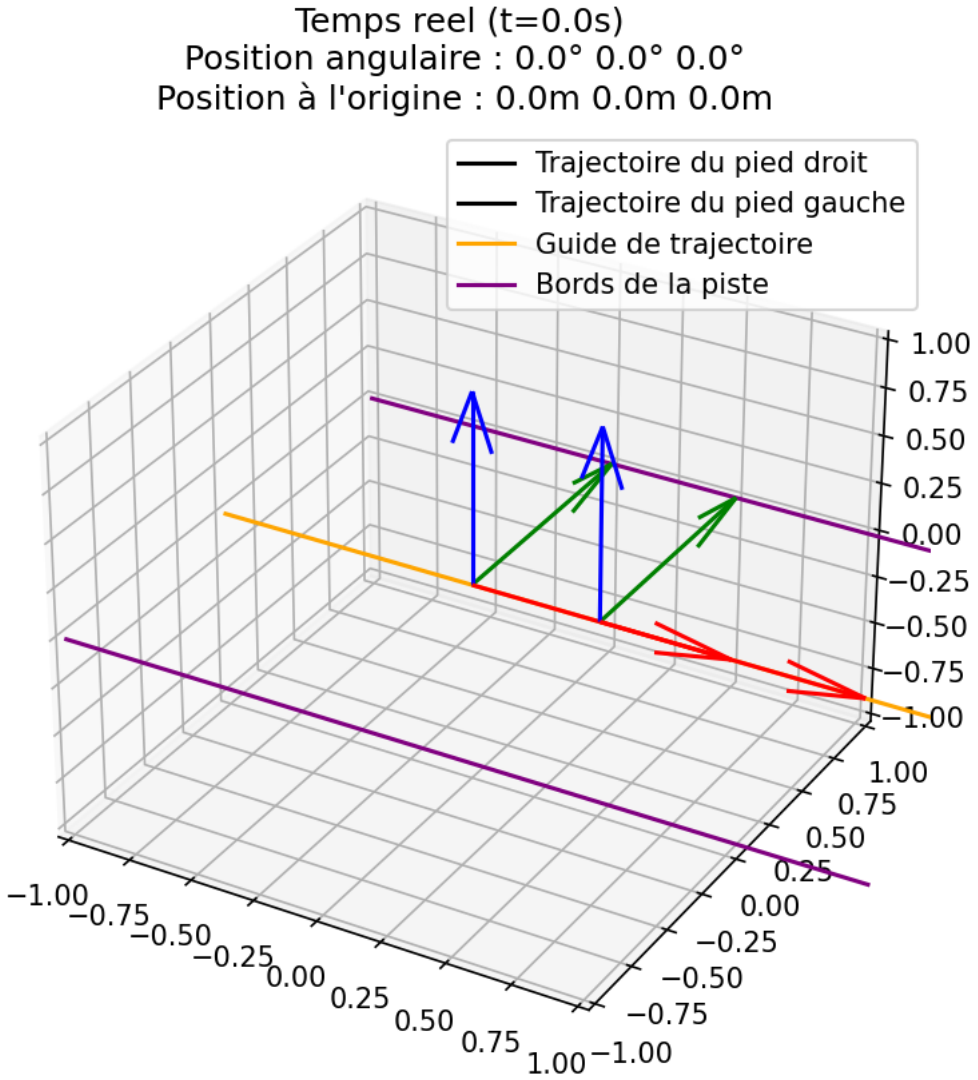
Résultats de visualisation



Coordonnées en mètres et repère à l'instant t (essai intermédiaire)

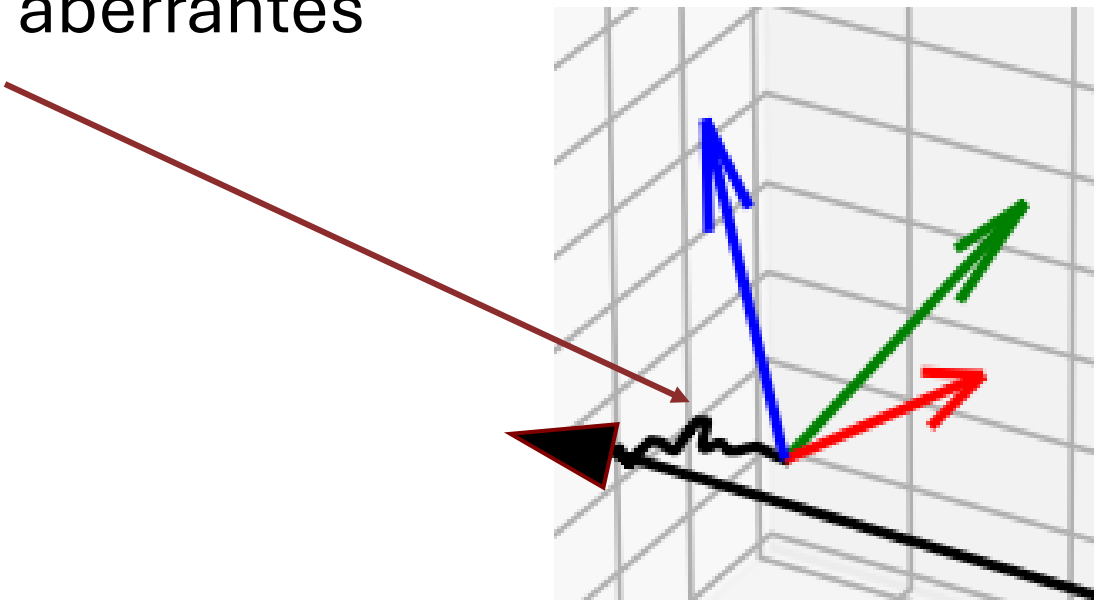
Utilisation de la visualisation

Système complet de visualisation et le patineur en position de départ, en travers de la piste



Ecarts

- Technologies imparfaites
- Dérive des données brutes des capteurs
- Trajectoires aberrantes



Améliorations possibles

- Fiabilisation des données
- Ajout de la position du buste
- Affichage de la trajectoire idéale simulée

Conclusions

- Réponse aux objectifs du MCOT
- Réponse à la problématique
- Projet améliorable
- Amélioration de la gestuelle