

# Rapport d'analyse des données ARE 2025

Lou Eouzan

19 mars 2025

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Imports</b>	<b>1</b>
<b>3</b>	<b>Ouverture du fichier et extraction des colonnes</b>	<b>2</b>
<b>4</b>	<b>Obtention de l'allure de la montée et de la descente</b>	<b>2</b>
<b>5</b>	<b>Tentative d'obtention de la vitesse</b>	<b>3</b>
<b>6</b>	<b>Tentative d'obtention du temps d'escalade</b>	<b>5</b>
<b>7</b>	<b>Calcul du ratio temps de mouvements sur temps total</b>	<b>5</b>
<b>8</b>	<b>Tentative d'obtention de la hauteur du mur</b>	<b>6</b>
<b>9</b>	<b>Conclusion</b>	<b>6</b>

## 1 Introduction

Lors de l'avant dernière séance nous avons réalisé une prise vidéo que nous avons du analyser à l'aide du logiciel kinovéa. Via le logiciel, nous procédons au tracking de la positions sur un axe x (horizontal) et un axe y (vertical) d'un point correspondant approximativement au centre de gravité tout du long de la vidéo. L'objectif de ce rapport est d'abord d'analyser les mesures obtenues pour en déduire certaines données comme le rapport entre le temps de mouvement et le temps d'immobilité, la vitesse d'escalade, le temps d'escalade et la hauteur du mur mais également de comparer l'efficacité de ce mode de mesure avec celui utilisée lors du rapport 1 où nous avons mesuré l'accélérations linéaire sans g.

## 2 Imports

```
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
```

Ce code importe des bibliothèques python dont nous avons besoin. La bibliothèque matplotlib nous permet de produire des graphiques pour avoir une représentation visuelle des données. Enfin les bibliothèque numpy et pandas nous mettent à disposition des structures de données et des fonctions associées pour stocker et analyser nos mesures.

### 3 Ouverture du fichier et extraction des colonnes

```
data = pd.read_csv("data_traj_escalade_v1.5.csv", sep=";", decimal=",")
time = data["Time"]
x = data["Marqueur 1/0/X"]
y = data["Marqueur 1/0/Y"]
```

La première ligne importe les données dans une variable `data`, ensuite les lignes suivantes stockent chaque colonne du dataframe obtenu en lisant les données dans une variable correspondante. Les variables `x` et `y` stockent la position de notre point sur l'axe horizontal et vertical. Enfin la variable `time`, elle contient les secondes correspondant au moment où chaque valeur de position a été mesurée.

### 4 Obtention de l'allure de la montée et de la descente

```
print(np.where(data["Marqueur 1/0/Y"] == np.max(data["Marqueur 1/0/Y"])))
plt.clf()
plt.plot(x[:3433], y[:3433])
plt.xlabel("Position horizontale (en pixel)")
plt.ylabel("Position verticale (en pixel)")
plt.savefig("traj_montee.svg")
plt.clf()
plt.plot(x[3433:], y[3433:])
plt.xlabel("Position horizontale (en pixel)")
plt.ylabel("Position verticale (en pixel)")
plt.savefig("traj_descente.svg")
```

```
(array([3429, 3430, 3431, 3432]),)
```

La première ligne renvoie la liste des index pour lesquelles la positions sur l'axe vertical est la plus haute. Cette position la plus haute correspond logiquement au haut du mur et donc à l'index de la fin de la montée. En l'occurrence, le dernier index contenant la position la plus haute est l'index 3432. Pour afficher la trajectoire lors de la montée et la descente on veut afficher la position sur l'axe `x` et l'axe `y` respectivement de l'index 0 à l'index 3432 et de l'index 3433 à la fin du tableau. C'est ce dont se charge les lignes suivantes.

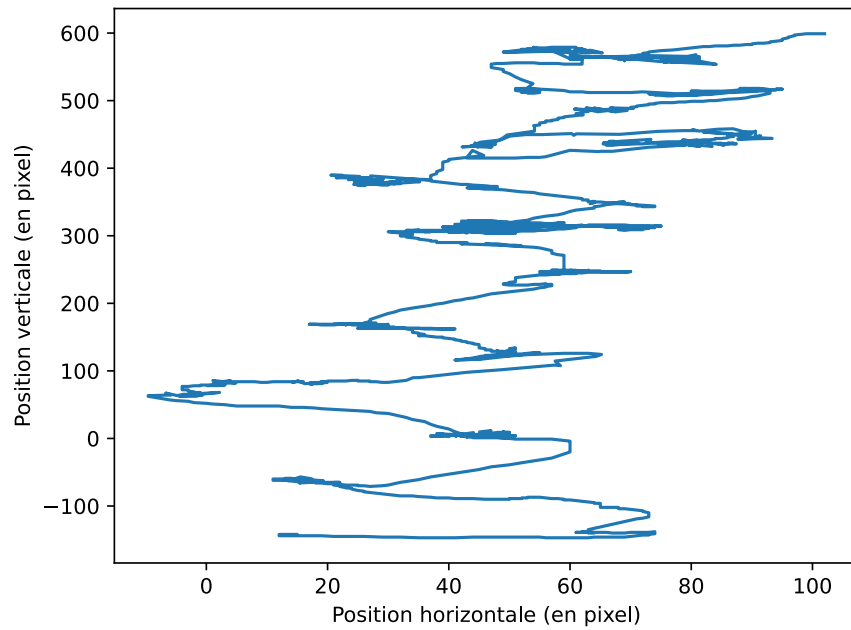


Fig. 1 : Tentative pour déterminer l'allure de la montée.

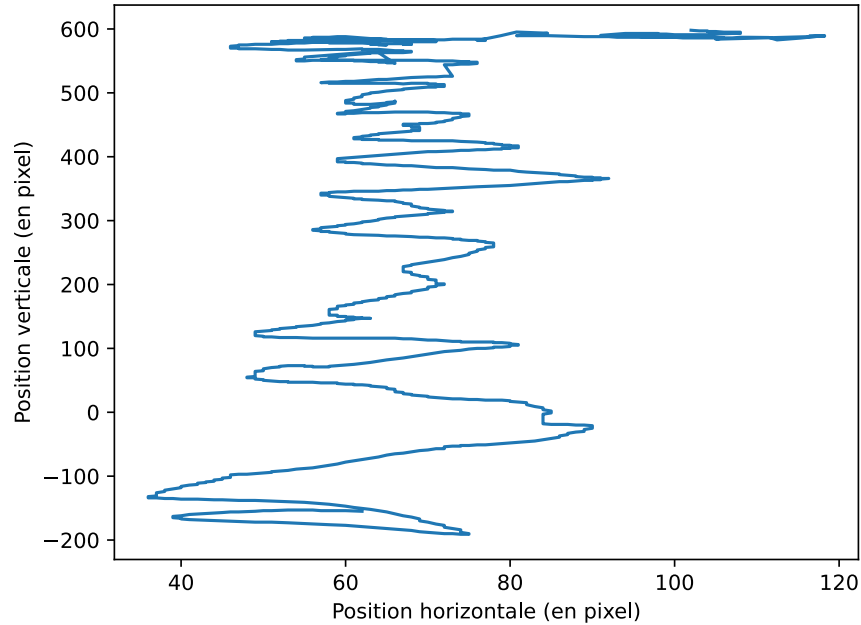


Fig. 2 : Tentative pour déterminer l'allure de la descente.

## 5 Tentative d'obtention de la vitesse

```
i = 0  
vitesses_par_seconde = []
```

```
# 3402 = fin de la montée-30
while i <= 3402:
    time_a, x_a, y_a = data.iloc[i]
    time_b, x_b, y_b = data.iloc[i+30]
    dist_en_cm = (np.sqrt((x_b-x_a)**2+(y_a-y_b)**2))/1.6542
    vitesses_par_seconde.append(dist_en_cm/(time_b-time_a)) # vitesse en cm/s
    i+=30

print(np.mean(np.array(vitesses_par_seconde))*0.036) # vitesse moyenne en km/h
```

0.2778923219981999

Pour connaître la vitesse en mètre par seconde, il est nécessaire de convertir les distances en mètre, pour cela a été pris la mesure de la distance entre mes pieds et le point de référence au début de la vidéo sur kinovéa. Cette distance s'est avéré être de 165,42 pixels correspondant après mesure avec un mètre à 100 centimètre. Nous pouvons donc obtenir par un simple produit en croix que  $1\text{cm} = 1 * 165,42/100$  pixels ce qui donne 1,6542 pixels. Nous devons donc diviser chaque distance par 1,6542 pour obtenir le résultat en cm. Le temps quand à lui est en secondes. La différence de temps entre chaque mesures est, d'après une simple lecture des premières données d'environ 0,033 secondes, il faut environ 30 mesures ( $1/0,033$ ) pour avancer d'une seconde. Les positions dont nous mesurons la distance correspondent donc à celle de la mesure  $x$  et de la mesure  $x + 30$ . Il suffit ensuite de faire une moyenne des vitesses mesurées lors de la montée en centimètres par secondes. Nous tentons en ensuite de convertir la vitesse en km/h,  $1\text{km} = 100000\text{cm}$  donc on commence par diviser par 100 000, par ailleurs 1 heure = 3600 secondes donc on multiplie par 3600, on a donc  $1\text{cm/s} = (1/100000) * 3600 = 0,036\text{km/h}$ , il suffit donc de multiplier les cm/s par 0,036 pour obtenir la vitesse en km/h. La vitesse obtenue est donc d'environ 0,28 km/h. Dans la mesure où, sur la vidéo nous pouvons voir que je suis assez lente et que par ailleurs je fais beaucoup de pause et j'hésite à monter, cette vitesse moyenne pourrait être cohérente. Elle l'est en tout cas plus que celle trouvée lors du premier rapport qui mesurait jusqu'à 10 000 m/s ( $= 10000 * 3,6 = 36000\text{km/h}$  (on divise par 1000 pour que ce soit en km et on multiplie par 3600 pour que ce soit en heure ce qui revient à faire  $x * (3600/1000) = x * 3,6$ )) ce qui était totalement invraisemblable.

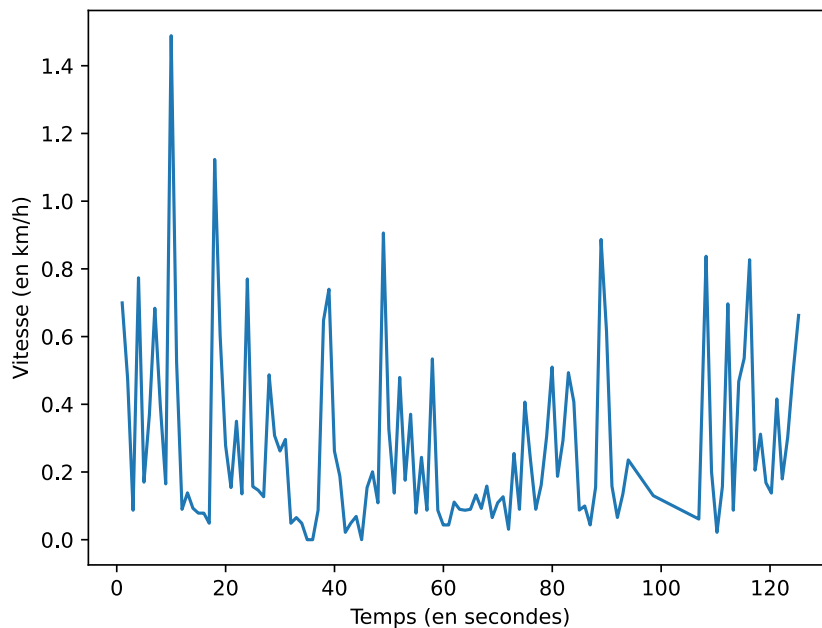


Fig. 3 : Vitesse mesurée au cours du temps lors de la montée.

## 6 Tentative d'obtention du temps d'escalade

```
print(time[3432])
```

125.648

Pour mesurer le temps d'escalade il suffit tout simplement de regarder la valeur à l'index 3432 du tableau time correspondant au temps passé depuis le début quand nous avons atteint le point le plus haut sur l'axe vertical, c'est à dire le haut du mur. Que ce soit pour ce rapport où le rapport précédent, la mesure du temps était cohérente.

## 7 Calcul du ratio temps de mouvements sur temps total

```
def isclose(a, b, trsh):
    return abs(a-b)<trsh
trsh = 1
index_deb = time[(isclose(time, 11.23, trsh))]
index_fin = time[(isclose(time, 17.33, trsh))]
while len(index_deb) > 1:
    trsh/=10
    index_deb = time[(isclose(time, 11.23, trsh))]
trsh = 1
while len(index_fin) > 1:
    trsh/=10
    index_fin = time[(isclose(time, 17.33, trsh))]
index_deb = index_deb.index[0]
index_fin = index_fin.index[0]
print(index_deb, index_fin)
```

337 520

Pour trouver les temps d'immobilités, nous avons identifiés visuellement entre 11.23 secondes et 17.33 secondes un temps d'immobilité. Nous allons donc chercher les index correspondant le plus à ces timecode puis mesurer la distance moyenne considérée comme parcourue sur cet interval et considérer ensuite que toute distance parcourue supérieure à cette distance moyenne correspond à du mouvement. C'est cette recherche dont se charge ce code en retrecissant le nombre d'élément autour de ces timecodes jusqu'à ne trouver plus que l'élément le plus proche.

```
i = 0
dist_imo = []
while i < len(data[index_deb:(index_fin+1)])-30:
    time_a, x_a, y_a = data.iloc[i]
    time_b, x_b, y_b = data.iloc[i+30]
    dist_imo.append((np.sqrt((x_b-x_a)**2+(y_a-y_b)**2))/1.6542)
    i+=30
trsh = np.mean(np.array(dist_imo))
print(trsh)
```

11.916358817123998

Ce code ci se charge de calculer la distance moyenne parcouru sur cet interval pour pouvoir considéré que toute période où la distance parcouru est supérieure ou égale à celle ci sera considérée comme une période de mouvement.

```

i = 0
mouvement = []
while i <= (len(time)-30):
    time_a, x_a, y_a = data.iloc[i]
    time_b, x_b, y_b = data.iloc[i+30]
    dist_en_cm = (np.sqrt((x_b-x_a)**2+(y_a-y_b)**2))/1.6542
    if dist_en_cm > trsh:
        mouvement.append(True)
    else:
        mouvement.append(False)
    i+=30
mouvement_def = []
for j in mouvement:
    for k in range(30):
        mouvement_def.append(j)
if i < len(time):
    diff = len(time)-i
    for j in range(diff):
        mouvement_def.append(mouvement_def[-1])
print(f"round(((len(time[(np.array(mouvement_def))])*0.033)-0.033)/(time[3432])*100, 2)} %")

```

43.13 %

Pour chaque intervalle de 1 seconde environ où on considère qu'il y a du mouvement, on ajoute True dans le tableau mouvement pour l'indiquer sinon on ajoute false. Ensuite, nous créons un 2e tableau mais où chaque élément est présent 30 fois pour correspondre aux 30 index sautés pour faire nos mesures sur un intervalle de 1 seconde. Enfin on vérifie que le tableau définitif contient bien le même nombre d'éléments que **time** et si ce n'est pas le cas, on ajoute le dernier élément du tableau. C'est nécessaire car soit un tableau numpy quelconque `arr` et un tableau de booléen, `arr[(cond)]` ne renvoie que les éléments de `arr` pour lequel `cond` est égal à True, en l'occurrence correspondant aux moments où on est en mouvement. Chaque éléments de ce tableau correspondants à 0,033 secondes on multiplie la taille du tableau obtenu par 0,033 puis on soustrait une fois 0,033 puisque au premier élément du tableau 0 secondes se sont passées. On répète le même processus avec tous le tableau du temps. On fait ensuite le rapport entre les 2 chiffres obtenus puis on multiplie par 100 pour obtenir le pourcentage. Nous obtenons donc 33.43%, ça paraît peu dans la mesure où dans le dernier rapport avait été mesuré 45.54% de temps en mouvements alors même que je m'étais stoppée pendant de longue période à plusieurs reprises. La méthode avec accéléromètre semble donc plus efficace pour mesurer l'absence ou non de mouvement.

## 8 Tentative d'obtention de la hauteur du mur

```
print((np.mean(np.array(vitesses_par_seconde))/100)*(time[3432]))
```

9.699059576230507

Nous savons que  $vitesse = distance / temps$ , ça signifie donc que  $distance = vitesse * temps$ . Nous avons la vitesse en cm/s, il suffit de la diviser par 100 pour obtenir la vitesse en mètre par secondes. Nous multiplions la vitesse par le temps total d'escalade et on obtient un résultat d'environ 9,6 mètres. Ce résultat paraît réaliste. Lors du dernier rapport il n'avais pas été possible de calculer la hauteur du mur dans la mesure où la vitesse n'était pas exploitable, l'utilisation de kinovea et du trackage est donc plus efficace pour cette mesure.

## 9 Conclusion

Dans ce rapport, nous avons mesuré la positions de mon centre de gravité dans un plan. De cette position et de son évolutions à travers le temps nous avons obtenus la vitesse, le temps d'escalade, le ratio temps de mouvements sur temps total et la hauteur du mur. Cette méthode à globalement été plus efficace et précise que celle de la mesure à l'aide de l'accéléromètre.