Projet de Recherche
Mathématiques Appliquées
2019/2020

Détection de pas dans des signaux physiologiques

Réalisé par :
Yassine FILALI
Encadrant:
M. Charles TRUONG

Promotion 2021
Enseignant référent:
M. Zacharie ALES

Stage éffectué du 18/05/2019 au 14/08/2019

Organisme d'accueil : Centre Borelli ENS Paris-Saclay
Adresse : 4, avenue des Sciences 94235 Cachan Cedex

Résumé

Ce rapport présente une méthode de détection de pas dite "offline" basée principalement sur la détection des points de changement, utilisée sur une base de signaux biologique issus d'une expérience réalisée avec la collaboration de différents organismes médicaux avec plus de 200 patient et au total plus de 1000 données prélevées. Différentes approches sont abordées pour cette détection, et les résultats sont présentés pour chacune d'entre elles, avec certaines qui ont une précision de détection supérieure à 95%.

Mots-clés : Détection de pas, ruptures, traitement de signaux biomédicaux, analyse de la marche, Machine learning.

Abstract

This report presents an offline method for step detection mainly using change-point detection, tested on a database of biomedical signals from an experiment realised with the collaboration of different medical organisms, with more than 200 patient and 1000 data recorded. Different approaches are tested, with the results provided for each one of them, and some of them working with a good precision of detection which exceeds 95%

Keywords : Step detection, ruptures, biomedical signals processing, gait analysis, machine learning.

Remerciements

Je tiens à exprimer ma sincère gratitude et ma reconnaissance particulière à mon encadrant, M. Charles TRUONG qui m'a proposé ce projet, et qui par ses conseils constructifs m'a guidé tout au long de ce stage pour en faire une expérience très enrichissante pour moi.
Je tiens aussi à remercier M. Zacharie ALES d'avoir accepté d'être mon enseignant référent et d'avoir répondu à mes demandes quand j'en avais besoin.
Enfin, je remercie toute personne qui a participé d'une façon ou d'une autre à l'avancement de ce projet.

Table des matières

Table des figures

Liste des tableaux

Introduction

La marche est un mécanisme complexe qui est constitué d'une succession d'étapes bien précises. Certaines maladies peuvent engendrer une modification dans la marche humaine et ainsi, leurs faire perdre de l'autonomie et augmenter les risques de chute. Ainsi, la marche est directement liée à plusieurs pathologies comme la maladie de Parkinson, certaines formes de cancer..., ce qui donne encore plus d'importance à la nécessité de quantifier la marche, et de pouvoir l'analyser afin d'améliorer la détection de ces pathologies. Cette quantification se fait essentiellement à l'aide de capteurs qui ont connu un développement conséquent ces dernières années, rendant la prise de mesures relativement plus faciles. Plusieurs travaux ont tenté de quantifier la marche, principalement à l'aide de capteurs IMU (Inertial Measurement Units) composés de gyroscopes, d'accéléromètres et de magnétomètres, qui sont faciles d'utilisation, peu coûteux et ne nécessitement pas de préparations préliminaires.

Un élément central de la marche est le pas, et dont la quantification permettra une meilleure analyse de ce mécanisme. La structure d'un pas est différente selon l'état de santé de la personne, c'est pour cela qu'il est nécessaire de développer une méthode de qui permet une détection avec une bonne précision quelque soit l'état pathologique de la personne.

Différentes approches de détection de pas ont été développées par plusieurs outils. Dans ce projet, on a principalement utilisé la détection de points de changements pour assurer cette détection. La structure du rapport est comme suit : Le premier chapitre a pour but de présenter les données et le protocole expérimental pour les prélever. Le second chapitre présente les notions centrales de la détection de points de changement. Enfin, le troisième et dernier chapitre détaille l'approche utilisée pour assurer la détection de pas ainsi que les résultats obtenus.

Chapitre 1

Présentation des données

Les données principalement utilisées pour ce stage ont été récoltées entre Avril 2014 et Octobre 2015, sur des patients dans plusieurs institutions médicales. Les données ont été prélevées sur 3 groupes de patients classés selon qu'ils soient sains, atteints d'une pathologie neurologique, ou une pathologie orthopédique. Au total, Le nombre de patients considérés dans cette étude est de 230 : 52 patients sains, 125 patients atteints d'une pathologie neurologique et 53 atteints d'un trouble orthopédique. La principale référence pour cette partie est l'article [1].

Groupe pathologique	Nombre de patients	Nombre d'essais	Sexe M/F	Age(années)	Taille(cm)	Poids(kg)
Sain	52	242	35/17	36.4 (20.6)	173.4 (10.8)	70.7 (12.2)
Orthopédique	53	243	26/27	60.1 (19.3)	169.2 (10.2)	77.7 (16.8)
Neurologique	125	535	80/45	61.5 (13.2)	169.8 (8.7)	72.7 (15.6)
Total	230	1020	141/89	55.5 (19.6)	170.5 (9.7)	73.4 (15.4)

TABLE 1.1 – Tableau récapitulatif des patients ayant été mesurés pour l'expérience, la moyenne et (l'écart type) sont affichés.

1.1 Protocole de prélèvement des données :

Les données sont naturellement prélevées de la même manière pour chaque patient et chaque essai, à l'aide de deux capteurs IMU (Inertial Measurement Units)[1] placés aux deux pieds et qui mesurent l'accélération et la vitesse angulaire pour chaque pied. Les signaux sont prélevés en effectuant une séquence bien précise d'actions[1] :

— Rester immobile pendant 6 secondes.

— Marcher 10 mètres sans contrainte sur la vitesse.

— Faire un demi-tour en U.

— Remarcher jusqu'au point de départ.

— Rester immobile pendant 2 secondes.

Le prélèvement des données a été fait sans assistance aux patients et avec la vitesse naturelle de chacun. En moyenne, le protocole a duré 30.1 secondes, pour une durée maximale de 186.4 secondes et une durée minimale de 11.7 secondes[1].

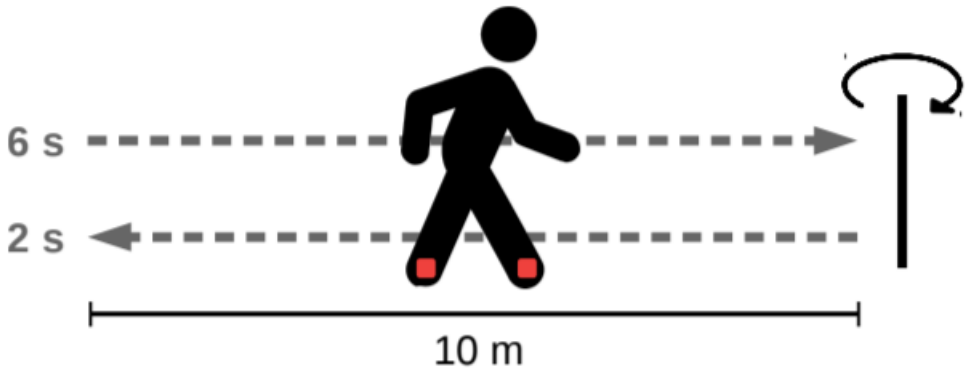

FIGURE 1.1 – Schéma descriptif du protocole de l'expérience[1]

1.2 Description des données

Les données prélevées pour un patient et un essai donné sont de deux types : Des séries temporelles illustrant les grandeurs mesurées par les capteurs, et des meta-données décrivant des informations relatives à l'expérience en cours.

1.2.1 Séries temporelles

Les séries temporelles prélevées par chaque capteur sont respectivement les projections de l'accélération et de la vitesse angulaire pour chaque pied, projetées sur les axes X,Y,Z et V, ou les axes X,Y et Z constituent le repère attaché aux capteurs, tandis que l'axe V est parallèle à la verticale, c'est à dire à la gravité comme montré dans la figure 1.2 . Ainsi, chaque instant de la série temporelle est un vecteur de $\mathbb{R}^{16}$.

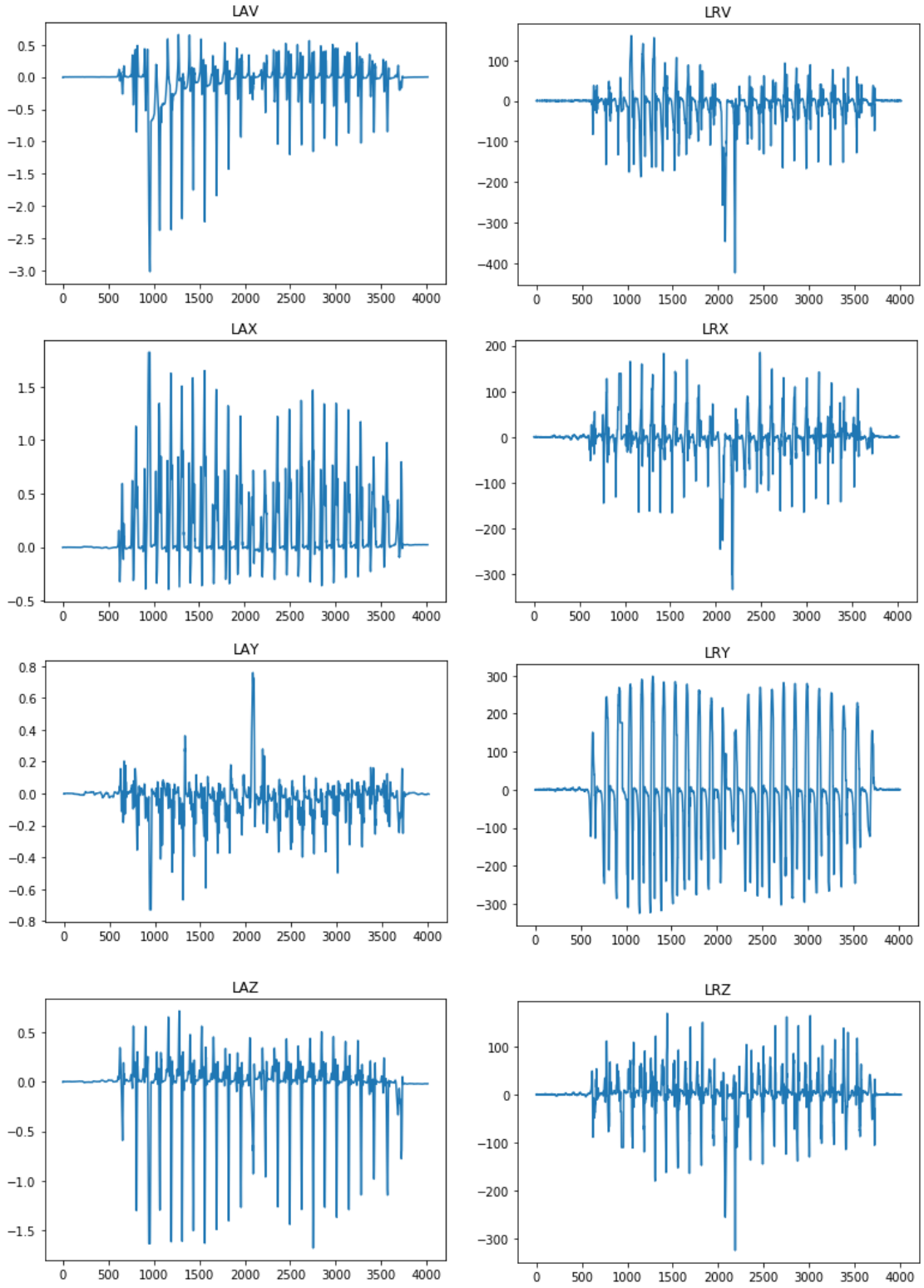

FIGURE 1.2 – Exemple des signaux enregistrés pour l'activité du pied gauche pour un essai

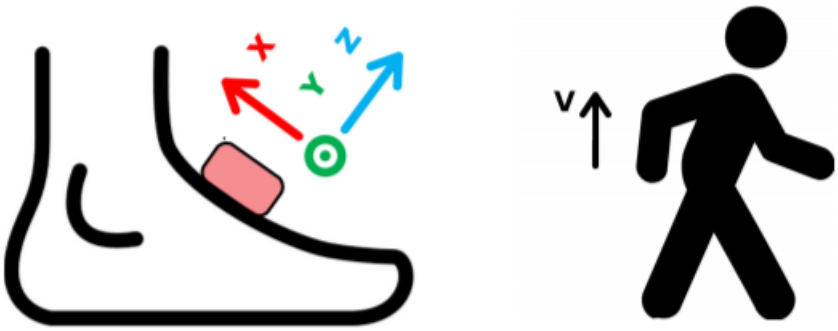

FIGURE 1.3 – Illustration détaillant l'orientation des axes considérés[1]

1.2.2 Meta-données

Les meta-données sont des données additionnelles à celles des séries temporelles qui sont en rapport avec le patient et l'expérience en cours. Ces données contiennent 16 variables au total, parmi lesquelles on a le code de l'expérience, les modèles des capteurs utilisés, des informations sur le patient (Age, IMC, pathologies ...) et enfin les périodes d'activité des pieds gauches et droits. On s'intéressera principalement à l'activité des pieds afin d'évaluer la précision de notre détection de pas. D'après les spécialistes, une marche est une succession de cycles composées de 4 étapes successives : heel-strike (HS), toe-strike (TS), heel-off (HO) and toe-off (TO). Une période d'activité d'un pied est mesurée entre les étapes HO et HS.

Pour chaque expérience, les signaux sont stockés dans des fichiers *csv* tandis que les meta-données sont stockées dans des fichier *json*, Avec pour code "numéro_du_partient"-"numéro_de_l'essai"."extension".

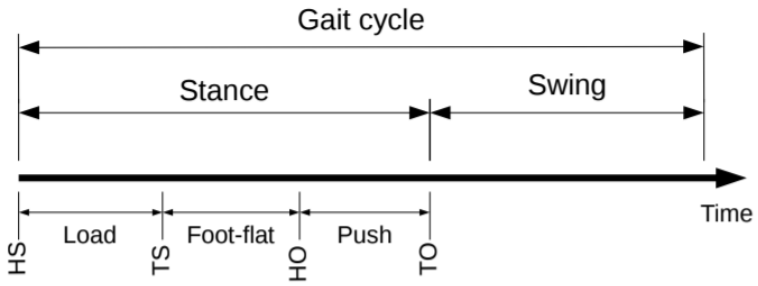

FIGURE 1.4 – Différentes étapes du cycle de la marche[1]

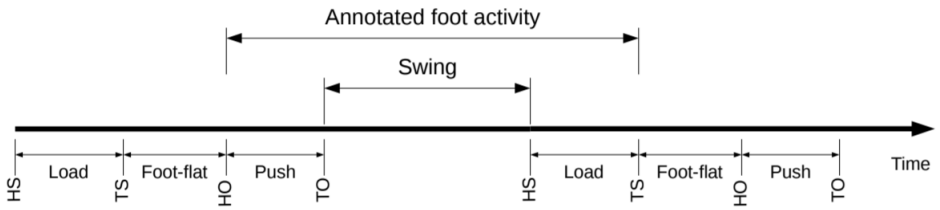

FIGURE 1.5 – Détails sur l'annotation d'une forte activité d'un pied[1]

Chapitre 2

Détection de ruptures

L'outil mathématique principal utilisé pour détecter les pas dans les signaux introduits dans la partie qui précède est la détection de points de changement, qui consiste à repèrer dans une série temporelle les points où le modèle du signal (paramétrique ou pas) est susceptible de changer. Les premiers travaux ont commencé par la détection d'un changement de moyenne dans un échantillon de variables aléatoire identiquement distribuées suivant une loi normale. Cette méthode de détection de points de changements est dite "offline", puisque les changements sont détectés en considérant le signal dans sa totalité. D'autres approches dites "online" sont possibles, par exemple par des méthodes bayesiennes [3] .

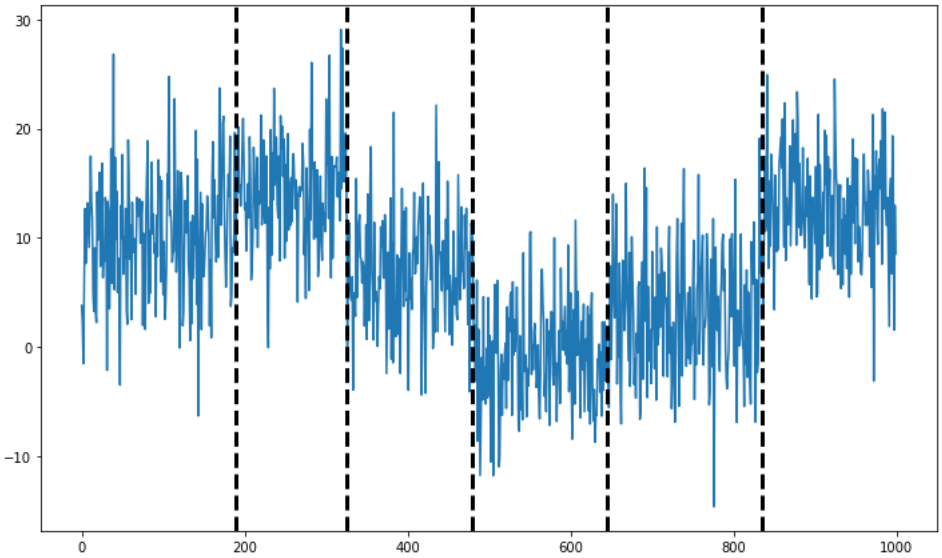

FIGURE 2.1 – Exemple d'une détection de points de changement sur un signal présentant un changement dans la moyenne de sa distribution.

2.1 Position du problème

Considérons un signal $y = \{y_t\}_{t=1}^{T}$ à valeurs dans $\mathbb{R}^d$, avec $d \geq 1$. On suppose que ce signal est stationnaire par morceau, c'est à dire qu'il existe des instants $t_1 < t_2 < ... < t_N$ à partir desquels le comportement du signal change d'une certaine manière (Un changement d'un paramètre de sa distribution par exemple). La détection de points de changement consiste à estimer les instants $t_1 < t_2 < ... < t_N$. Cela peut se formuler comme le choix d'une ségmentation $\tau = \{t_1, t_2, ..., t_N\}$ qui minimise un certain critère $\chi(\tau)$. On considère aussi que le critère χ est une sommation de **fonctions de coût**[2].

Ainsi on a que $\chi(\tau) = \sum_{k=0}^{N} c(y_{t_k, t_{k+1}})$, $t_k \in \tau$, avec $y_{t_{k+1}, t_k} = \{y_t\}_{t=t_k}^{t_{k+1}-1}$ un sous-signal de $y = \{y_t\}_{t=1}^{T}$. On distingue deux types de problèmes de minimisation suivant le cas ou le nombre de points de changements est connu ou pas :

— **Problème 1 : Nombre de points de changements connu :** Lorsque le nombre de points de changements est connu au préalable, disons N, On considère la ségmentation τ dont le cardinal $|\tau|$ est égal à N, qui minimise le critère $\chi(\tau)$. Le problème de minimisation dont le suivant :

$$\min_{|\tau|=N} \chi(\tau)$$

— **Problème 2 : Nombre de points de changements quelconque :** Pour un nombre de points de changement quelconque, une fonction de pénalisation sur la ségmentation $pen(\tau)$ est introduite , Ainsi le problème d'optimisation pour ce cas est comme suit :

$$\min_{\tau} \chi(\tau) + pen(\tau)$$

Ainsi, la recherche de points de changements dans un signal repose sur 3 notions centrales[2] :

— **La fonction de coût** utilisée pour calculer le critère à minimiser. Elle permet de détecter les changements dans le comportement du signal, suivant le modèle choisi : Elle prend des valeurs faibles quand le sous-signal est homogène, et des valeur élevée s'il y a des changements des le comportement du signal (section 2.2).

— **La méthode de résolution** C'est l'algorithme utilisé pour la résolution des problèmes d'optimisation décrit ci-dessus (section 2.3).

— **La fonction de pénalisation** Dans le cas d'une détection sans connaissance au préalable du nombre de points de changements, une fonction de pénalisation pour obtenir le problème de minisation numéro 2 (section 2.4).

2.2 Fonctions de coût

Dans cette section, sont présentées les fonctions de coût qui seront utilisées dans la partie suivante pour la détection de pas. Comme mentionné précédemment, deux types de

fonction de coût peuvent être distinguées, celles appartenant aux modèles paramétriques et celles aux modèles non-paramétriques. Chaque choix de fonction de coût permet la détection des points de changement selon un critère particulier.

2.2.1 Modèles paramétriques

Les modèles paramétriques se focalisent sur les changements qui surviennent sur des paramètres multidimensionnels. Parmi ce type de fonctions de coût, on considère la famille de coûts par maximum de vraisemblance. Pour cela, le signal y est supposé identiquement distribué par morceaux, c'est à dire qu'à un instant t donné, la densité de y_t s'écrit comme suit[2] :

$$y_t \sim \sum_{i=0}^{N} f(.|\theta_i) 1_{\{t_i \leq t \leq t_{i+1}\}}$$

Ou les t_i, $i \in \{1, N\}$ sont les instants de changement du signal, avec la convention $t_0 = 0$ et $t_{N+1} = T$. Ainsi, les changements dans le signal se font sur les différentes valeurs θ_i. En considérant ces hypothèses, le critère à minimiser peut être vu comme une maximisation de la vraisemblance sur le signal : On voudrait estimer le paramètre θ_i sur chaque sous ségment par maximum de vraisemblance, et donc on adopte une fonction de coût, en passant à la log-vraisemblance, de la forme :

$$c_{MLH}(y_{a,b}) = -\max_{\theta} \sum_{i=a}^{b-1} Log(f(y_i|\theta))$$

la densité f peut être associée à différentes lois, dans notre cas on suppose que f est la densité d'une loi normale multivariée d'espérance $\mu \in \mathbb{R}^d$ et de matrice de covariance $\Sigma \in \mathcal{M}_d(\mathbb{R})$. Ainsi on la densité suivante :

$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{N/2} det(\boldsymbol{\Sigma})^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{y_i} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{y_i} - \boldsymbol{\mu})\right]$$

Ainsi la fonction de coût devient :

$$c_{normal}(y_{a,b}) = -\max_{\theta} \sum_{i=a}^{b-1} Log(f(y_i|\theta))$$

$$= -\max_{\mu,\Sigma} \sum_{i=a}^{b-1} Log(\frac{1}{(2\pi)^{N/2} det(\boldsymbol{\Sigma})^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{y_i} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{y_i} - \boldsymbol{\mu})\right])$$

$$= -\max_{\mu,\Sigma} \sum_{i=a}^{b-1} -\frac{N}{2} Log(2\pi) - \frac{1}{2} det(\Sigma) - \frac{1}{2}(\boldsymbol{y_i} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{y_i} - \boldsymbol{\mu})$$

Les estimateurs de μ et Σ par maximum de vraisemblance coïncident avec les estimateurs empiriques de l'espérance et de la matrice de covariance, qu'on note respectivement $\hat{\mu_{a,b}}$ et $\hat{\Sigma_{a,b}}$. On aura donc :

$$c_{normal}(y_{a,b}) = \frac{N(b-a)}{2} Log(2\pi) + \frac{(b-a)}{2} det(\hat{\Sigma_{a,b}}) + \frac{1}{2} \sum_{i=a}^{b-1} (\boldsymbol{y_i} - \boldsymbol{\hat{\mu_{a,b}}})^\top \boldsymbol{\hat{\Sigma_{a,b}}}^{-1} (\boldsymbol{y_i} - \boldsymbol{\hat{\mu_{a,b}}})$$

Avec l'expression des deux estimateurs :

$$\hat{\mu_{a,b}} = \frac{1}{b-a} \sum_{i=a}^{b-1} y_i$$

$$\hat{\Sigma_{a,b}} = \frac{1}{b-a} \sum_{i=a}^{b-1} (y_i - \hat{\mu_{a,b}})(y_i - \hat{\mu_{a,b}})^T$$

Comme le premier terme $\frac{N(b-a)}{2} Log(2\pi)$ n'apporte rien aux problèmes de minimisation, puisqu'il engendrera un terme constant qui sera toujours présent égal à $\frac{(T+1)N}{2} Log(2\pi)$, On peut le supprimer de la fonction de coût. Pour finir, une fonction de coût qui a l'expression suivante est obtenue[2] :

$$c_{normal}(y_{a,b}) = (b-a) \det(\hat{\Sigma}_{a,b}) + \sum_{i=a}^{b-1} (\boldsymbol{y_i} - \boldsymbol{\hat{\mu_{a,b}}})^\top \boldsymbol{\hat{\Sigma_{a,b}}}^{-1} (\boldsymbol{y_i} - \boldsymbol{\hat{\mu_{a,b}}})$$

Même si cette fonction de coût est basée sur la densité d'une loi normale multidimensionnelle, elle permet de détecter les changement dans l'espérance et la matrice de covariance de plusieurs distributions autre que gaussiennes[2].

2.2.2 Modèles non-paramétriques

Les fonctions de coûts non-paramétriques sont des fonctions qui font l'hypothèse que le signal est distribué selon une loi qui n'est pas nécessairement paramétrique, donc que :

$$y_t \sim \sum_{i=0}^{N} F_i \; 1_{\{t_i < t \leq t_{i+1}\}}$$

Ou les F_i sont des densités de lois qui ne sont pas forcément paramétriques. Parmi cette famille de fonctions de coût, quelques fonctions de coût basées sur les noyaux (kernel-based methods) sont présentées.

Définition 2.2.1. Noyau (Kernel) défini positif (Théorème d'Aronszajn)[4] :
$\mathcal{K} : \chi \times \chi \to \mathbb{R}$ une fonction symétrique, est dite noyau défini positif sur l'ensemble χ si et seulement si il éxiste un espace de Hilbert $\mathcal{H}$ et une fonction $\phi : \chi \to \mathcal{H}$ tels que :

$$k(x, x') =< \phi(x), \phi(x') >_{\mathcal{H}} \qquad \forall \;\; x, x' \in$$

Définition 2.2.2. Espace de Hilbert à noyau reproductible (RKHS)[4] :
Soit χ un ensemble et $\mathcal{H} \subset \mathbb{R}^{\chi}$ un sous espace des fonctions de $\chi \to \mathbb{R}$ muni de son produit scalaire $< ., . >_{\mathcal{H}}$. Une fonction $K : \chi \times \chi \to \mathbb{R}$ est appelée **noyau reproductible** de $\mathcal{H}$ si elle vérifié les propriétés suivantes :

— $\mathcal{H}$ contient toutes les fonctions de la forme :

$$\forall x \in \chi, K_x : t \longmapsto K(x,t)$$

— $\forall x \in \chi, \forall f \in \mathcal{H}$:

$$f(x) =< f, K_x >_{\mathcal{H}}$$

Si une telle fonction K existe, $\mathcal{H}$ est appellé **Espace de Hilbert à noyau reproductible.**

Propriétés[4] : Quelques propriétés des noyaux reproductibles et des RKHS :

— Si $\mathcal{H}$ est un RKHS, alors il admet un unique noyau reproductible.

— Inversement, si K est un noyau reproductible, il ne peut l'être que pour au plus un RKHS.

— Tout noyau défini positif est un noyau reproductible.

Ainsi, On se place dans un RKHS $\mathcal{H}$, son noyau reproductible $k(.,.) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ et la fonction $\phi : \mathbb{R}^d \to \mathcal{H}$ qui lui est associée. L'image du signal par ϕ est $\phi(y_t)$. L'idée pour aboutir à la fonction de coût est de calculer la somme des carrés résiduels dans l'espace H. La moyenne sur un sous signal $y_{a,b}$ est définie comme suit[5] :

$$\hat{\mu_{a,b}} = \frac{1}{b-a} \sum_{i=a}^{b-1} \phi(y_i)$$

Ainsi la somme des carrés résiduels (non normalisée) sur $y_{a,b}$ est :

$$c_{kernel}(y_{a,b}) = \sum_{i=a}^{b-1} ||\phi(y_i) - \hat{\mu_{a,b}}||^2_{\mathcal{H}}$$

$$= \sum_{i=a}^{b-1} < \phi(y_i) - \hat{\mu_{a,b}}, \phi(y_i) - \hat{\mu_{a,b}} >_{\mathcal{H}}$$

$$= \sum_{i=a}^{b-1} < \phi(y_i), \phi(y_i) >_{\mathcal{H}} -2 < \hat{\mu_{a,b}}, \phi(y_i) >_{\mathcal{H}} + < \hat{\mu_{a,b}}, \hat{\mu_{a,b}} >_{\mathcal{H}}$$

— $< \hat{\mu_{a,b}}, \hat{\mu_{a,b}} >_{\mathcal{H}} = \frac{1}{(b-a)^2} \sum_{i=a}^{b-1} \sum_{j=a}^{b-1} < \phi(y_i), \phi(y_j) >_{\mathcal{H}}$

— $2 < \hat{\mu_{a,b}}, \phi(y_i) >_{\mathcal{H}} = \frac{2}{b-a} \sum_{j=a}^{b-1} < \phi(y_i), \phi(y_j) >_{\mathcal{H}}$

On a donc :

$$c_{kernel}(y_{a,b}) = \sum_{i=a}^{b-1} < \phi(y_i), \phi(y_i) >_{\mathcal{H}} - \frac{2}{(b-a)} \sum_{i=a}^{b-1} \sum_{j=a}^{b-1} < \phi(y_i), \phi(y_j) >_{\mathcal{H}}$$

$$+ \frac{1}{(b-a)^2} (b-a) \sum_{i=a}^{b-1} \sum_{j=a}^{b-1} < \phi(y_i), \phi(y_j) >_{\mathcal{H}}$$

$$= \sum_{i=a}^{b-1} < \phi(y_i), \phi(y_i) >_{\mathcal{H}} - \frac{1}{(b-a)} \sum_{i,j=a}^{b-1} < \phi(y_i), \phi(y_j) >_{\mathcal{H}}$$

Enfin comme on a que $k(x,y) =< \phi(x), \phi(y) >_{\mathcal{H}}$, on aboutit à la fonction de coût de la forme suivante[6] :

$$c_{\text{kernel}}(y_{a,b}) = \sum_{i=a}^{b-1} k(y_i, y_i) - \frac{1}{b-a} \sum_{i,j=a}^{b-1} k(y_i, y_j)$$

Avec $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ la fonction noyau utilisée pour la construction de ce coût. Plusieurs noyaux sont utilisables, et le choix dépend surtout de la nature du signal. Dans ce projet, deux noyaux seront utilisés :
Le noyau linéaire : $k(x,y) = \langle x, y \rangle_{\mathbb{R}^d}$ avec $x, y \in \mathbb{R}^d$ On aboutira à une fonction de coût :

$$c_{\text{lin}}(y_{a,b}) = \sum_{i=a}^{b-1} ||y_i||^2 - \frac{1}{b-a} \sum_{i,j=a}^{b-1} < y_i, y_j >_{\mathbb{R}^d}$$

Ce noyau est utilisé pour détecter les changements de moyenne dans le signal.
Le noyau gaussien[6] : $k(x,y) = \exp(-\gamma \|x-y\|^2)$ et $x, y \in \mathbb{R}^d$ et $\gamma > 0$ La fonction de coût obtenue en utilisant le noyau gaussien devient :

$$c_{\text{rbf}}(y_{a.b}) = (b-a) - \frac{1}{b-a} \sum_{i,j=a}^{b-1} \exp\left(-\gamma \|y_i - y_j\|^2\right)$$

2.3 Méthode de résolution

Les méthodes de resolution sont les algorithmes qui peuvent être utilisés afin de résoudre les problèmes de minimisation discrêts, que ce soit pour un nombre connu ou inconnu de points de changement. Il existe deux type d'algorithmes, des méthodes qui fournissent une solution éxacte des problèmes de minimisation, et des méthodes approximatives qui fournissent des solutions approximatives. Après différents essais de plusieurs algorithmes (Programmation dynamique, fenêtre glissante, Binary segmentation et BottomUp), seul l'algorithme BottomUp sera utilisé car il offre un bon compromis en terme de rapidité d'exécution et de précision du résultat[2].

L'algorithme BottomUp : Le principe de l'algorithme est de commencer avec le plus de points de changement, suivant un paramètre $\delta > 2$ qui représente l'épaisseur d'un ségment pour les points de changements initialisés, et d'en supprimer les moins pertinents selon une certaine métrique qui associée à la fonction de coût à chaque itération. L'algorithme s'arrête lorsque la condition d'arrêt, qui est spécifiée selon la nature du problème de minimisation (avec ou sans pénalisation) est satisfaite. La complexité de cet algorithme est de l'ordre de $\mathcal{O}(n \log n)$[7], ou n est le nombre d'échantillons, c'est à dire la taille du signal. Un pseudo-code de l'algorithme est présenté ci-dessous. Les conditions d'arrêt de l'algorithme dépendent du type de problème :

— Si c'est un problème à nombre de points de changement connu, l'algorithme s'arrête lorsque le nombre de points de changement est atteint.

— Si c'est un problème avec nombre de points de changement inconnu, l'algorithme s'arrête lorsque la partie pénalisation ($pen(\tau)$) du critère sur la ségmentation devient supérieure au critère ($\chi(\tau)$).

Algorithme 1 Algorithme BottomUp[2]

Entrée : le signal $\{y_t\}_{t=0}^{T}$, la fonction de coût $c(.)$, la condition d'arrêt, δ l'épaisser de l'initialisation.

Initialisation : $L \leftarrow \{\delta, 2\delta, \ldots, (\lfloor T/\delta \rfloor - 1)\delta\}$ Liste des points de changements initiaux.

Répèter

-$k \leftarrow |L|$

-$t_0 \leftarrow 0$ et $t_{k+1} \leftarrow T$

-Annoter par $t_i (i = 1, \ldots, k)$ les éléments de L par ordre ascendant, tel que $L = \{t_1, t_2, ...t_k\}$

-Initialiser G une liste de longueur $k - 1$

Pour i de 1 jusqu'à $(k - 1)$ **faire**

$-G[i-1] \leftarrow c\left(y_{t_{i-1}t_{i+1}}\right) - \left[c\left(y_{t_{i-1}t_i}\right) + c\left(y_{t_i,t_{i+1}}\right)\right]$

Fin Pour

$-\hat{i} \leftarrow \text{argmin}_i\, G[i]$

-Retirer t_{i+1} de L

Jusqu'à Condition d'arrêt satisfaite

Sortie : L liste des points de changements estimés

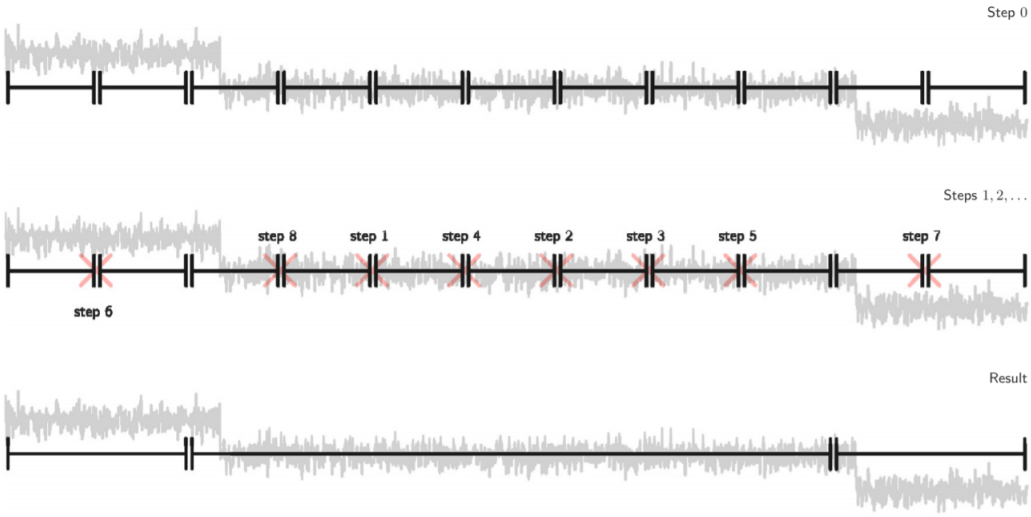

FIGURE 2.2 – Schéma du principe de l'algorithme BottomUp[2]

2.4 Fonctions de pénalisations

Plusieurs types de fonctions de pénalisation peuvent être utilisées pour une détection de points de changement sans connaître leur nombre. Parmi ces pénalisations, des pénalisations linéaires seront principalement utilisées.
Une fonction de pénalisation linéaire s'écrit sous la forme[2] :

$$pen(\tau) = \beta|\tau|$$

Où $\beta \in \mathbb{R}, \beta > 0$ est un paramètre à calibrer et $|\tau|$ le cardinal de la ségmentation. Pour des valeurs faibles de β, des changements plutôt fréquents sont détectés, tandis que pour des grandes valeurs de β, la plupart des points de changement ne sont pas détectés. Ci-dessous un exemple illustrant l'effet d'un calibrage de β sur la détection des points de changement.

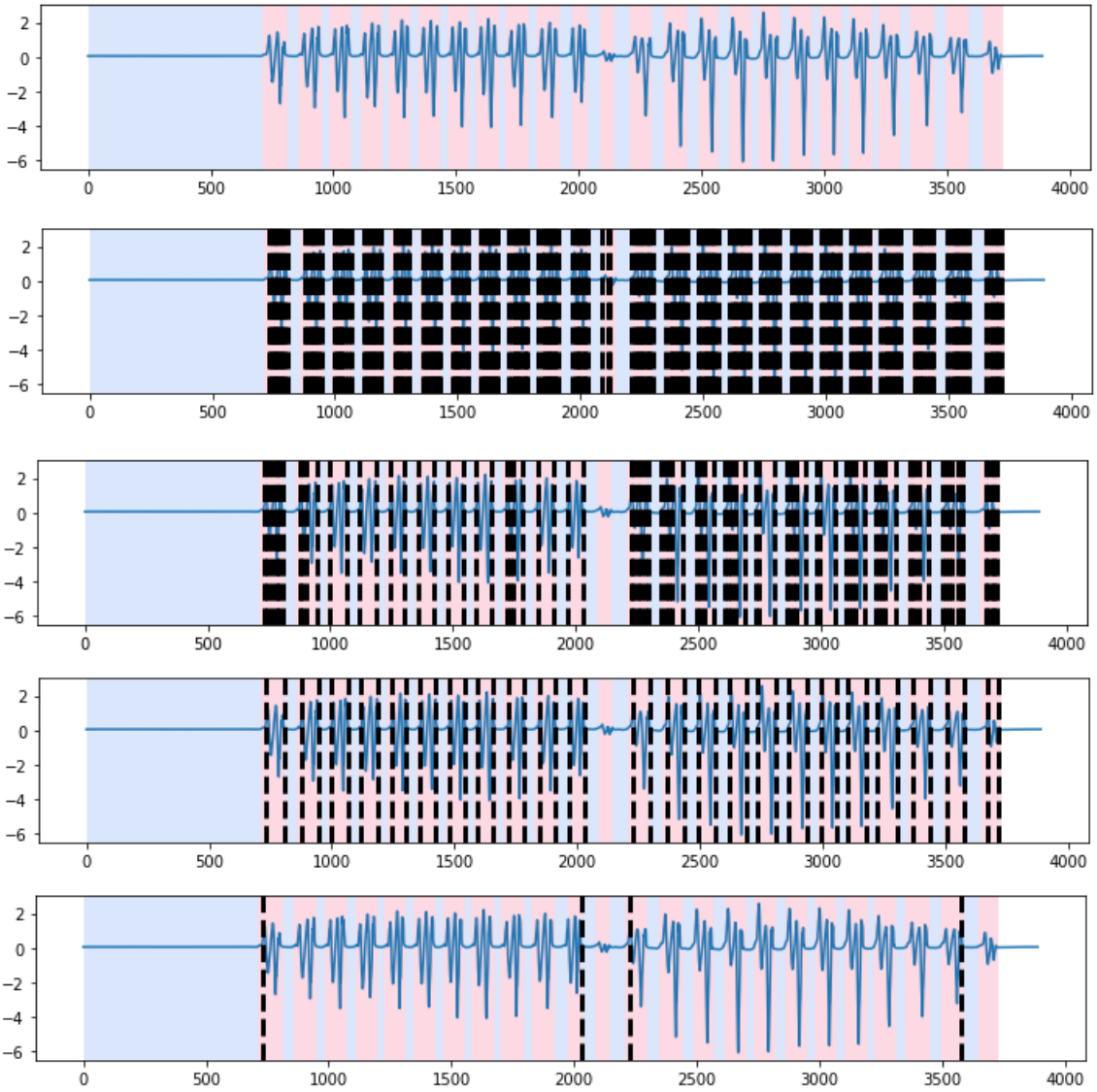

FIGURE 2.3 – Signal d'origine et détection de changements pour des valeurs croissantes du paramètre β

Chapitre 3

Extraction des pas

Dans cette partie, l'application de la détection des ruptures pour détecter des pas sera présenter. Cette procédure peut être décortiquée en 3 grandes étapes :

— Ségmenter le signal.

— Identifier les pas.

— Evaluer le modèle.

On commence tout d'abord par une détection avec connaissance au préalable du nombre de pas, puis une pénalisation est introduite afin d'adapter la méthode pour des situations où le nombre de pas est inconnu.

Le principal outil de travail est le langage Python, avec les bibliothèques scientifiques et de Machine Learning usuelles, ainsi que le package **Ruptures**[7] qui offre un moyen efficace d'implémenter les fonctions de coût et les différents éléments de la détection de points de changements.

3.1 Métriques d'évaluation

Ici sont introduites les métriques d'évaluation utilisée car ce sera le principal critère pour évaluer la précision de la détection. 3 métriques sont principalement utilisées qui sont **la précision**, **le rappel (recall)**, et **le F1-score**[8].
l'ensemble des vrais pas pour un essai est noté par $\mathcal{P}$, le i-ème vrai pas par P_i, son début par $t_{i,s}$ et sa fin par $t_{i,e}$, les pas détectés auront la même notation avec une $(*)$ au dessus.

3.1.1 Précision

Un pas détecté P_i^* est compté correct si la moyenne de son instant de début $t_{i,s}^*$ et de son instant de fin $t_{i,e}^*$ est située dans l'intervalle d'un vrai pas P_j qui est $[t_{j,s}, t_{j,e}]$, autrement dit :

$$\exists j \in \{1, .., |\mathcal{P}|\}, \frac{t_{i,s}^* + t_{i,e}^*}{2} \in [t_{j,s}, t_{j,e}]$$

Si plusieurs pas détectés sont comptés corrects par rapport au même vrai pas, seul un d'entre eux est comptabilisé. En notant l'ensemble des pas détectés corrects par P_c^*, la précision est définie par :

$$precision = \frac{|P_c^*|}{|P^*|}$$

.

3.1.2 Rappel

Un vrai pas P_i est considéré comme détecté avec succès si la moyenne de son instant de début $t_{i,s}$ et de son instant de fin $t_{i,e}$ est située dans l'intervalle d'un pas détecté P_j^* qui est $[t_{j,s}^*, t_{j,e}^*]$, autrement dit :

$$\exists j \in \{1, .., |\mathcal{P}*|\}, \frac{t_{i,s} + t_{i,e}}{2} \in [t_{j,s}^*, t_{j,e}^*]$$

Si plusieurs vrais pas sont comptés comme détectés avec succès par rapport au même pas détecté P_k^*, seul un d'entre eux est comptabilisé. En notant l'ensemble des vrais pas détectés avec succès par P_d, le rappel est défini par :

$$rappel = \frac{|P_d|}{|P|}$$

. On peut remarquer un rôle symétrique entre le rappel et la précision.

3.1.3 F1-score

Le F1-score est défini en fonction de la précision et du rappel par la formule :

$$F1 - score = 2 \times \frac{precision \times rappel}{precision + rappel}$$

C'est la métrique principale qui servira à évaluer la détection de pas.

3.2 Pré-processing

Cette partie concerne principalement les calculs préliminaires et les traitements qui sont effectués sur les données, soit pour accélérer les calculs, soit pour améliorer la précision de détection.

3.2.1 Calculs préliminaires pour la fonction de coût

Afin d'avoir un calcul du coût sur un ségment de complexité $\mathcal{O}(1)$, des classes adaptées disponibles avec Ruptures sont utilisée, qui possèdent deux méthodes :

— Une méthode **fit** qui effectue des calculs préliminaires.

— Une méthode **error** qui calcule le coût sur un sous signal $y_{a,b}$.

En prenant comme exemple la fonction de coût c_{lin} :

$$c_{\text{lin}}\left(y_{a,b}\right) = \sum_{i=a}^{b-1} ||y_i||^2_{\mathbb{R}^d} - \frac{1}{b-a} \sum_{i,j=a}^{b-1} < y_i, y_j >_{\mathbb{R}^d}$$

$$= \sum_{i=a}^{b-1} ||y_i||^2_{\mathbb{R}^d} - \frac{1}{b-a} < \sum_{i=a}^{b-1} y_i, \sum_{j=a}^{b-1} y_j >_{\mathbb{R}^d}$$

$$= \sum_{i=a}^{b-1} ||y_i||^2_{\mathbb{R}^d} - \frac{1}{b-a} || \sum_{j=a}^{b-1} y_j ||^2_{\mathbb{R}^d}$$

Pour ce cas, deux somme cumulées sont précalculées, une somme cumulée sur les $||y_i||^2_{\mathbb{R}^d}$ pour le premier terme et une somme cumulée sur les y_i pour le second terme. Ainsi, le coût sur le sous-signal $y_{a,b}$ est calculé directement en effectuant des différences en utilisant ces sommes cumulées.

3.2.2 Scaling

Puisque les différentes variables ne sont pas du même ordre de grandeur, où les variables liées à la vitesse angulaire ont un ordre de grandeur plus important que celles liées à l'accélération, il est nécessaire de se ramener au même ordre de grandeur afin d'appliquer certains modèles.

```
In [5]: d.iloc[:,4:8].describe().iloc[1:,:]
Out[5]:
              RRV         RRX         RRY         RRZ
mean    -3.015206   -7.182806    0.273112   -1.583578
std     38.688181   39.575133  123.237977   51.788069
min   -358.843800 -265.800000 -333.300000 -266.900000
25%     -2.508607   -7.000000  -12.500000   -8.900000
50%     -0.319311   -1.200000    0.300000    0.100000
75%      2.307908    1.000000    4.400000    2.800000
max    157.121700  186.200000  300.100000  242.700000
```

```
In [4]: d.iloc[:,0:4].describe().iloc[1:,:]
Out[4]:
              RAV         RAX         RAY         RAZ
mean    -0.012216   0.151455    0.009014    0.011542
std      0.254052   0.380738    0.144125    0.242364
min     -1.571206  -0.624647   -0.349060   -1.342032
25%     -0.011956  -0.000847   -0.040561   -0.009732
50%      0.000129   0.011053   -0.002161    0.000968
75%      0.038478   0.163753    0.010039    0.100268
max      0.638124   1.939453    0.931040    0.658068
```

FIGURE 3.1 – Mise en évidence de la différence d'ordre de grandeur entre l'accélération et la vitesse angulaire

Ainsi en rendant toutes les variables centrées réduites, on ramène toutes les variables au même ordre de grandeur.

3.2.3 Sélection des variables

Ces méthodes sont toutes testées avec le même algorithme après avoir transformé les données et avec une fonction de coût c_{normal}. Le F1-score en considérant toutes les variables est égal à 0.9382.

Sélection par corrélation Pour cette méthode, les variables qui ont une corrélation supérieure à un certain seuil sont groupées. Par exemple si une variable V1 est très corrélée à une variable V2 qui elle même est très corrélée à une variable V3, ces variables forment un groupe, et une seule variable de ce groupe est considérée. En essayant pour un seuil de corrélation égal à 0.5 (Car en moyenne, la valeur maximale de la corrélation entre 2 variables distinctes est de l'ordre de 0.7), on obtient un F1-score égal à 0.9027. Cette méthode n'est pas très efficace car les variables ne sont pas très corrélées entre elles.

PCA En essayant une réduction de dimension par PCA en considérant un nombre de composantes qui exprime 95% de la variance pour chaque fichier. On obtient un F1-score égal à 0.9225. Cette méthode est peu efficace car le signal n'est pas de grande dimension (dimension 8).

Accélération ou Vitesse angulaire Comme mentionné dans le premier chapitre, les principaux signaux tracés sont liés à l'accélération ou à la vitesse angulaire. En considérant seulement l'accélération, on obtient un F1-score égal à 0.9352, et 0.9216 en considérant seulement la vitesse angulaire. Le résultat obtenu pour seulement les variables d'accélération est très proche du résultat initial en les considérant toutes.

3.3 Identification des pas

Plusieurs méthodes ont été utilisées pour pouvoir identifier la nature d'un ségment, si c'est un pas ou non. En premier lieu, un critère lié à l'écart type du ségment est utilisé pour déterminer sa nature. Ensuite, un clustering est effectué afin d'identifier les pas et enfin, on effectue une classification supervisée.
Pour cette partie, tous les tests seront effectués par une fonction de coût c_{normal}, avec nombre de pas supposé connu et scaling sur les signauxs.

3.3.1 Identification par écart type

Le critère d'identification d'un ségment dans ce cas est simplement qu'au delà d'un certain seuil limite de la norme de l'écart type. On considère que le ségment correspond à un pas. Le seuil limite considéré est l'écart-type du signal dans sa totalité car les période d'inactivité sont plus fréquentes sur les signaux et donc l'écart type sur un pas devrait être supérieur à l'écart type global.

Pathologie	Précision	Rappel	F1-score
Sain	0.9470	0.8756	0.9086
Neurologique	0.9545	0.8688	0.9083
Orthopédique	0.9607	0.9038	0.9301
Moyenne	0.9543	0.8788	0.9136

TABLE 3.1 – Résultats pour une identification de pas par écart-type.

3.3.2 Identification par clustering (k-moyennes)

La prochaine idée pour identifier la nature des segments est d'effectuer un clustering binaire sur les segments, pour cela, il est nécessaire de définir des descripteurs de ségments pour ensuite pouvoir faire un clustering par k-moyennes[9].

Deux descripteurs sont principalement utilisés, les deux issus du résultat de la fonction *describe()* du package *pandas*, qui retourne un tableau avec pour lignes la moyenne, l'écart type, le minimum, le maximum et les 3 premiers quartiles pour chacune des variables. Les signaux considérés sont ceux du pied droit.

```
In [10]: d.describe()
Out[10]:
                RAV           RAX  ...           RRY           RRZ
count  3890.000000  3890.000000  ...  3890.000000  3890.000000
mean     -0.014501     0.167316  ...     0.426452    -0.975887
std       0.276525     0.401716  ...   127.214136    58.831102
min      -1.703824    -0.588326  ...  -360.900000  -287.900000
25%      -0.018008     0.000074  ...   -17.600000   -11.075000
50%      -0.000106     0.013474  ...     0.000000     0.200000
75%       0.040544     0.225649  ...     3.000000     3.100000
max       0.701489     1.938774  ...   303.300000   211.700000

[8 rows x 8 columns]

In [11]: d.describe().shape
Out[11]: (8, 8)
```

FIGURE 3.2 – Exemple du retour de la fonction describe()

Ainsi les descripteurs utilisés sont :

— **Descripteur normé :** Un descripteur qui prend la norme des vecteurs lignes du tableau, dont un vecteur de dimension 7.

— **Descripteur aplati :** Un descripteur qui prend toutes les valeurs du tableau, mais "aplati" grâce à la fonction *numpy.flatten()*, donc un vecteur de dimension 7 × *le nombre de variables*.

Deux approches sont possibles pour le clustering, soit un clustering par fichier, qui forme les clusters a partir des segments d'un seul fichier, soit un clustering global qui extrait les descripteurs des segments de tous les fichiers pour ensuite faire un clustering global, et ensuite identifier les pas pour chaque fichier. Il est à noter que pour l'approche globale, on associe à chaque segment le fichier auquel il est associé afin de pouvoir le récupérer à la fin pour calculer les métriques d'évaluation pour chaque fichier.

Ensuite, après avoir effectué le clustering, on doit distinguer le cluster des pas de celui des non-pas. Comme par exemple, Le cluster ayant une valeur moyenne de la norme de l'écart-type supérieure est considéré comme celui des pas. Pour la suite, on considère que l'identification des clusters se fait correctement.

Pathologie	Précision	Rappel	F1-score
Sain	0.9461	0.9020	0.9223
Neurologique	0.9549	0.9191	0.9354
Orthopédique	0.9642	0.9198	0.9403
Moyenne	0.9551	0.9153	0.9335

TABLE 3.2 – Résultats pour une identification par k-moyennes, avec un descripteur aplati et un clustering par fichier.

Pathologie	Précision	Rappel	F1-score
Sain	0.9413	0.9137	0.9262
Neurologique	0.9463	0.9295	0.9369
Orthopédique	0.9598	0.9366	0.9472
Moyenne	0.9491	0.9266	0.9368

TABLE 3.3 – Résultats pour une identification par k-moyennes, avec un descripteur normé et un clustering par fichier.

Pathologie	Précision	Rappel	F1-score
Sain	0.9450	0.9202	0.9315
Neurologique	0.9569	0.9099	0.9314
Orthopédique	0.9628	0.9442	0.9528
Moyenne	0.9556	0.9205	0.9366

TABLE 3.4 – Résultats pour une identification par k-moyennes, avec un descripteur aplati et un clustering global.

Pathologie	Précision	Rappel	F1-score
Sain	0.9410	0.9240	0.9316
Neurologique	0.9482	0.9220	0.9340
Orthopédique	0.9578	0.9505	0.9536
Moyenne	0.9489	0.9293	0.9382

TABLE 3.5 – Résultats pour une identification par k-moyennes, avec un descripteur normé et un clustering global.

On peut noter que l'utilisation d'un descripteur normé donne des résultats légèrement meilleurs, et un clustering global est légèrement meilleur qu'un clustering par fichier.

3.3.3 Identification par classification supervisée (forêt aléatoire)

Pour cette partie, on considère des descripteurs aplatis pour les segments. L'avantage d'une classification supervisée est que contrairement à la méthode non supervisée, on peux directement identifier la classe du ségment avec la prédiction. Pour cela, les fichiers contenant les signaux sont divisés en des fichiers pour l'entrainement du modèle et des fichiers pour tester le modèle.

Les données utilisées pour l'apprentissage et la prédiction sont regroupées dans des tableaux contenant les descripteurs pour chaque vrai segment ainsi que l'étiquette du segment en questin (si c'est un pas ou non).

Le modèle utilisé est un classifieur par forêt aléatoire[9] prenant comme seul paramètre le nombre d'arbres qui est par défaut à 100. Ce paramètre n'a pas été changé car en vue du temps mis pour entrainer le modèle (assez rapide) et des résultat obtenus qui seront détaillés par la suite, il est tout à fait légitime de le conserver.

En sélectionnant aléatoirement 10 fichiers pour chaque type de pathologie (10 fichiers pour des patients sain, 10 pour des patients atteints d'une maladie neurologique et 10 pour des patients atteints d'une maladie orthopétique), c'est à dire en considérant 30 fichiers pour tester le modèle et 990 fichier pour tester le modèle, la modèle a une précision très proche de 100% (Aux alentours de 99.8%). On en déduit que le modèle n'a pas besoin d'énormément de fichier pour apprendre.

Ainsi, On peux entamer la détection de pas après avoir entraîné le modèle. Afin d'évaluer ce modèle et d'avoir un résultat qui reflète la réalité, on effectue une validation croisée sur 5 plis :

— Avant de commencer, on considère tous les fichiers, classés suivant le type de pathologie associé. Les groupes sont ensuite mélangés, et divisés en 5 parties. On obtient une liste de 5 éléments, ou le i-ème élément sont les fichiers qui seront considérés pour entraîner à la i-ème itération de la validation croisée.

— A chaque itération, on entraîne avec les éléments de la liste construite au début (i-ème élément pour entraîner à la i-ème itération), et on utilise ce classifieur pour prédire sur le reste des fichiers. Comme pour la classification par k-moyennes, on parcourt tous ces fichiers pour extraire les descripteurs pour tous les segments. Ensuite on effectue une prédiction pour chaque ségment pour identifier les pas et on calcule les métriques d'évaluation.

— Le résultat final de la validation croisée est obtenu en calculant la moyenne des résultats pour chaque itération.

Pathologie	Précision	Rappel	F1-score
Sain	0.9654	0.9684	0.9662
Neurologique	0.9489	0.9580	0.9525
Orthopédique	0.9607	0.9616	0.9605
Moyenne	0.9555	0.9613	0.9576

TABLE 3.6 – Résultats pour une identification par forêt aléatoire avec seulement accélération.

Pathologie	Précision	Rappel	F1-score
Sain	0.9408	0.9427	0.94136
Neurologique	0.9489	0.9521	0.9500
Orthopédique	0.9601	0.9578	0.9585
Moyenne	0.9496	0.9512	0.9500

TABLE 3.7 – Résultats pour une identification par forêt aléatoire en considérant toutes les variables.

3.4 Détection avec nombre de pas inconnu

Pour cette partie, on utilise une identification des pas par forêt aléatoire, et on ne considère que les variables d'accélération, avec un scaling.

On considère d'abord une fonction de coût gausienne c_{normal} Sans connaissance du nombre de pas, nous introduisons une pénalisation linéaire au problème d'optimisation pour détecter les points de changement. La valeur de la pénalisation sur une ségmentation τ est $pen(\tau) = \beta|\tau|$.

L'idéal est de calibrer le paramètre β par cross validation sur les données de test. Cependant, une telle procédure prendrait beaucoup trop de temps si on le faisait sur tous les fichiers. Pour cela, on considère un nombre réduit de fichiers (10 de chaque type aléatoirement parmi les fichiers de test) avec lesquels on trace l'évolution du F1-score sur une grille de valeurs de β allant de 20 à 1000 avec un pas de 20.

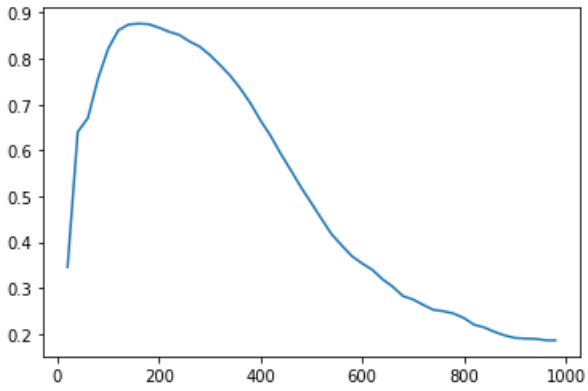

FIGURE 3.3 – Evolution du F1-score en fonction des valeurs du paramètre β

En repètant une telle procédure plusieurs fois, on obtient une valeur moyenne de β optimatle égale à 194 (Pour toutes les répétitions, le F-score maximal est entre 0.85 et 0.89). En testant avec cette valeur sur tous les fichiers par validation croisée, on obtient les résultats suivants.

Pathologie	Précision	Rappel	F1-score
Sain	0.8237	0.9450	0.8743
Neurologique	0.8462	0.8217	0.8263
Orthopédique	0.8284	0.9791	0.8927
Moyenne	0.8367	0.8876	0.8632

TABLE 3.8 – Résultats pour une détection de pas pour une pénalisation avec $\beta = 192$.

Il est à noter que le calibrage de β est coûteux en temps de calcul pour ne pas forcément donner un résultat très intéresant. En plus de cela, l'échelle de calibrage est différente en fonction du coût : Par exemple, pour un cout gaussien, un β optimal est de l'ordre de 200 dans notre cas avec seulement les mesures d'accélération (de l'ordre de 1000 si on considère toutes les variables) et de l'ordre de 10 pour des fonctions de coût à base de noyaux c_{kernel}. Enfin, le β optimal est choisi en moyenne, c'est à dire qu'il n'est pas forcément optimal pour tous les fichiers, mais sera proche de l'optimalité. Une approche différente est adoptée pour la suite.

On considère pour cela des coûts à base de noyaux à savoir c_{lin} et c_{rbf}. En considérant une constante de pénalisation $\beta = 1$, on obtient ce genre de résultat pour une détection de point de changement.

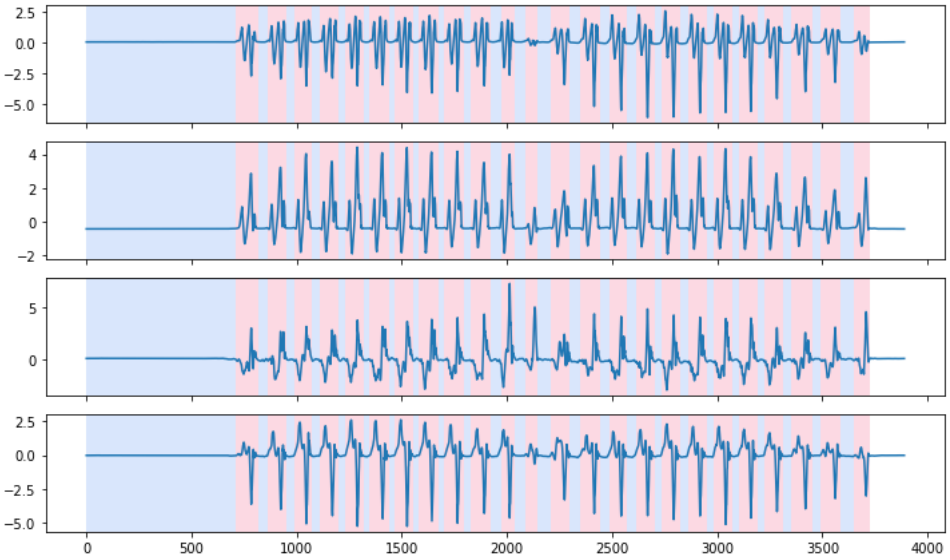

FIGURE 3.4 – Signaux d'origine

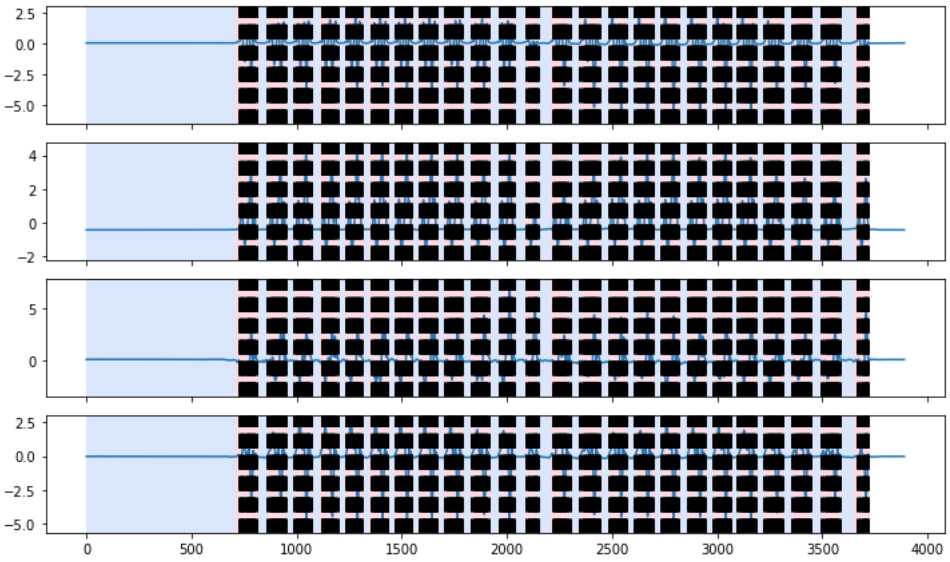

FIGURE 3.5 – Détection de pas pour $\beta = 1$ avec un coût c_{rbf}

Malgré la détection de changements avec un régime fréquentiel très rapide, on peut toujours distinguer l'allure des pas. On introduit une fonction pour ajuster les ségments,

qui concatène les segments adjacents d'une longueur inférieure à une longueur limite, en un seul ségment.

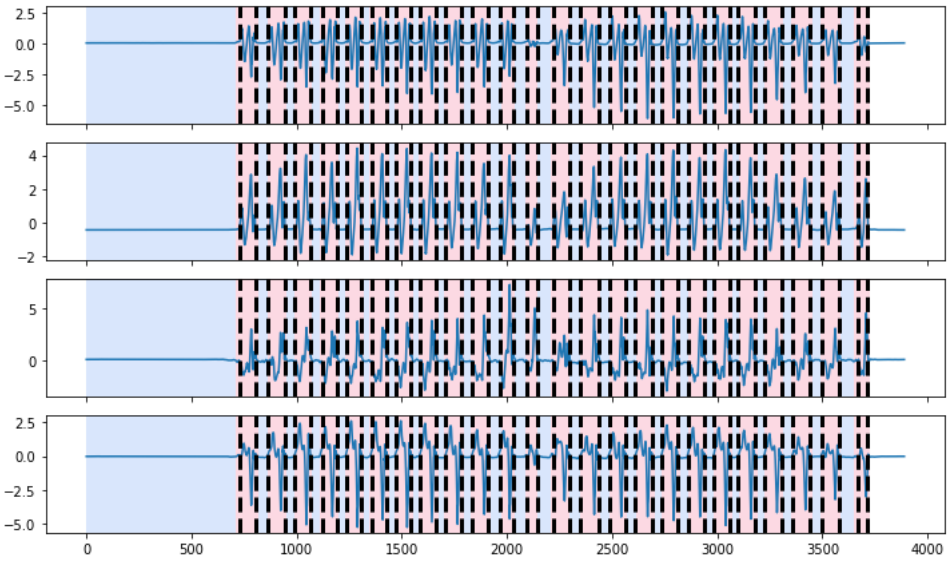

FIGURE 3.6 – Détection de pas pour $\beta = 1$ avec un coût c_{rbf} avec ajustement

Ainsi après ajustement, ont obtient une ségmentation des signaux bien meilleure. En testant sur tous les signaux par validation croisée, et avec les deux fonctions de coût c_{lin} et c_{rbf}, on obtient les résultats suivants.

Pathologie	Précision	Rappel	F1-score
Sain	0.9860	0.9876	0.9865
Neurologique	0.9767	0.9838	0.9797
Orthopédique	0.9810	0.9882	0.9842
Moyenne	0.9800	0.9858	0.9824

TABLE 3.9 – Résultats pour une détection de pas avec pénalisation et ajustement avec la fonction de coût c_{rbf}.

Pathologie	Précision	Rappel	F1-score
Sain	0.9734	0.9848	0.9785
Neurologique	0.9614	0.9778	0.9682
Orthopédique	0.9627	0.9780	0.9694
Moyenne	0.9646	0.9795	0.9709

TABLE 3.10 – Résultats pour une détection de pas avec pénalisation et ajustement avec la fonction de coût c_{lin}.

On peut noter que pour les résultats pour ces deux fonctions de coût sont bons, avec un performance meilleure avec c_{rbf}.

3.5 Récapitulatif

Ci-dessous un tableau récapitulatif des tests effectués.

Fonction de coût	descripteur	pénalisation	pré-processing	Identification de pas	F1-score
c_{rbf}	aplati	$\beta = 1$ + Ajust	Sc + Acc	RF	0.9824
c_{lin}	aplati	$\beta = 1$ + Ajust	Sc + Acc	RF	0.9709
c_{normal}	aplati	$\beta = 1$ + Ajust	Sc + Acc	RF	0.8757
c_{normal}	aplati	$\beta = 1$	Sc + Acc	RF	0.8632
c_{lin}	aplati	None	Sc + Acc	RF	0.6121
c_{rbf}	aplati	None	Sc + Acc	RF	0.9410
c_{normal}	aplati	None	Sc + Acc	RF	0.9576
c_{normal}	aplati	None	Sc + R	RF	0.9397
c_{normal}	aplati	None	Sc	RF	0.9500
c_{normal}	aplati	None	Sc	K-means	0.9500
c_{normal}	aplati	None	Sc	K-means	0.9366
c_{normal}	normé	None	Sc	K-means	0.9382
c_{normal}	normé	None	Sc	K-means	0.9382
c_{normal}	normé	None	Sc + PCA	K-means	0.9225
c_{normal}	normé	None	Sc + Acc	K-means	0.9352
c_{normal}	normé	None	Sc + R	K-means	0.9216
c_{normal}	normé	None	Sc + Std	K-means	0.9027
c_{normal}	None	None	Sc	Std	0.9136

TABLE 3.11 – Tableau récapitulatif des différents essais effectués.

Notations : Pénalisation :

— Ajust : Ajustement des segments.

Pré-processing ;

— Sc : Scaling

— PCA : Analyse de composante principale avec 95% de la variance expliquée.

— Acc : Prise en compte des variables d'accélération uniquement.
— R : Prise en compte des variables de vitesse angulaire uniquement.
— Std : Sélection par corrélation.

Identification de pas :

— RF : Identification par forêt aléatoire.
— K-means : Identification par K-moyennes.
— Std : Identification par écart-type

Conclusion

Au terme de ce projet, on a pu aboutir à des méthodes de détection de pas qui donnent une assez bonne précision de détection. Cependant, il se peut qu'il y ait encore moyen d'améliorer les résultats, par exemple en explorant d'autres fonctions de coût, de nouvelles méthodes de pré-processing ou utiliser des pénalisations plus complexes. Un autre point à améliorer aussi serait une optimisation des calculs numériques effectués.

Une prochaine étape pour ce projet serait peut être de pouvoir identifier l'étape à laquelle appartienne le pas, car certaines anomalies pathologique peuvent se faire remarquer exclusivement lors d'une étape précise du protocole expérimental adopté dans la prise des données, comment par exemple les débuts et fins de la phase de marche, le demi tour en U,... A titre d'exemple, les patients atteints de la maladie de Parkinson présentent un début de marche 40% plus lent que les autres patient[8].

Il aurait été aussi très intéressant d'essayer de tester l'application de cette méthode, dans l'automatisation d'une certaine tâche médicale, comme par exemple détecter un certain comportement lié à une maladie particulière en analysant les pas. Pour conclure, ce type de projet peut avoir des application très intéressantes et innovantes que ce soit dans le domaine médical mais pourquoi pas aussi être applicable dans d'autres domaines.

Bibliographie

[1] T. Moreau C. Provost A. Vienne-Jumeau A. Moreau2 P.-P. Vidal N. Vayatis S. Buffat A. Yelnik D. Ricard L. Oudre C. Truong, R. Barrois-Muller. A data set for the study of human locomotion with inertial measurements units. *Image Processing On Line*.

[2] Nicolas Vayatis Charles Truong, Laurent Oudre. Selective review of offline change point detection methods. *Elsevier Signal Processing*.

[3] David J.C. MacKay Ryan Prescott Adams. Bayesian online changepoint detection.

[4] Jean-Philippe Vert Julien Mairal. Machine learning with kernel methods.

[5] Zaıd Harchaoui Olivier Cappe. Retrospective mutiple change-point estimation with kernels.

[6] Sylvain Arlot, Alain Celisse, and Zaid Harchaoui. A kernel multiple change-point algorithm via model selection. *Journal of Machine Learning Research*, 20(162) :1–56, 2019.

[7] Charles Truong. Ruptures documentation.

[8] R. ; Moreau T. ; Truong C. ; Vienne-Jumeau-A. ; Ricard D. ; Vayatis-N. ; Vidal P.-P. Oudre, L. ; Barrois-Müller. Template-based step detection with inertial measurement units.

[9] Trevor Hastie Robert Tibshirani Hastie, Trevor and J H. Friedman. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction.*