Compte-rendu de mission Télécom Étude

Dispositif de contrôle de l'apnée du sommeil

Matthieu Carreau
Telecom Paris, Institut Polytechnique de Paris
F-91120, Palaiseau, France
matthieu.carreau@telecom-paris.fr

Juillet 2023

Table des matières

1 Utilsation du code python

1.1 Présentation du code

Le code est composé de quatre fichiers, un fichier principal *main.py*, et des fichiers contenant les fonctions nécessaires à la lecture, au traitement et à l'écriture des fichiers. Le fichier *csv_processing.py* contient les fonctions permettant de traiter les données des csv, il utilise les fonctions définies dans *read_write_csv.py* et *signal_processing.py*, qui permettent respectivement de lire et écrire les données, et d'appliquer les différents filtres sur les signaux.

1.2 Comment l'utiliser ?

Le code est écrit en python 3.8.10 et nécessite les trois librairies suivantes :
- *numpy*, version 1.26.0
- *matplotlib*, version 3.8.0
- *pandas*, version 2.1.0

Pour éxécuter le projet et traiter des fichiers csv, les quatre fichiers python doivent être contenus dans un même dossier, et seul le fichier *main.py* doit être éxécuté.

Il contient deux exemples correspondant aux deux cas d'usage. Le premier permet de traiter un unique fichier csv à l'aide de la fonction *cp.process_csv*, en précisant le chemin relatif vers le fichier csv original, et le chemin relatif vers le fichier de destination, à créer dans un dossier existant. Le second permet de traiter tous les fichiers csv d'un dossier à l'aide de la fonction *cp.process_folder*, en précisant le chemin relatif vers le dossier d'origine, et celui de destination. De nouveaux fichiers seront créés dans le dossier de destination avec les mêmes noms que les fichiers originaux.

2 Description des algorithmes utilisés

2.1 Lecture des fichiers

Afin de pouvoir lire les différents formatages possibles pour les fichiers csv, la première étape est de créer de nouveaux fichiers normalisés dont le nom se termine par *_norm*. Les fichiers csv contiennent différentes séquences séparées par des en-têtes donnant le nom de chaque colonne, anisi à la lecture, on sépare les différentes séquences écrites dans le fichier. On forme pour chaque séquence un *DataFrame* contenant toutes ses données.

2.2 Extraction des parties exploitables des fichiers

La première étape du traitement d'une séquence est la détermination des parties exploitables du signal. Pour cela, on s'intéresse aux données de rythme cardiaque et de saturation en oxygène, qui sont celles qui contiennent des valeurs manquantes, représentées par des valeurs nulles. On considère qu'une partie d'une colonne du fichier est exploitable si :

- Sa longueur est supérieure à une longueur minimale *min_length* (fixée à 15 dans les arguments de la fonction *extract_relevant_parts*) en nombre d'échatillons ,

- Les potentielles valeurs manquantes qu'elle contient ne forment jamais une suite de plus de *max_missing_data_length* échantillons consécutifs.

Après avoir identifié les parties exploitables de chaque colonne de saturation ou de rythme cardiaque de la séquence, on sélectionne uniquement les parties qui sont simultanément exploitables dans chacune de ces colonnes, les autres données seront ignorées.

2.3 Nettoyage des données de rythme cardiaque et de saturation en oxygène

Les données de rythme cardiaque et de saturation en oxygène subissent une même prémière étape de nettoyage. Les variations de ces données sont normalement relativement lentes par rapport à la période d'échantillonage de deux secondes, ainsi, il est pertinent de réduire le niveau de bruit qu'ils présentent grâce à des méthodes de filtrage. On détecte tout d'abord les valeurs aberrantes par rapport au type de données à l'aide d'un seuil bas et d'un seuil haut définis pour chaque type de données. Ces seuils sont définis (et modifiables) dans les arguments par défaut de la fonction *process_part*. Les valeurs dépassant ces seuils sont considérées aberrantes et sont remplacées par 0 de façon à être traitées de la même façon que les valeurs manquantes.

Ensuite, les valeurs manquantes sont interpolées linéairement à partir des valeurs voisines.

Cette étape de préfiltrage est réalisée par la fonction *prefilter_part* de *csv_processing.py*.

Comme les signaux restent très bruitées, une opération de filtrage est réalisée. Deus filtres différents sont utilisés selon le type de signal.

2.3.1 Rythme cardiaque : filtrage de Wiener

Pour le rythme cardiaque, on applique un filtre de Weiner, qui permet d'atténuer les variations du signal attribuées au bruit, tout en conservant

les variations attribuées au signal utile. Il repose sur des hypothèses sur les densités spectrales du bruit et du signal utile, et va atténuer plus fortement les fréquences pour lesquelles le bruit est supposé plus fort que le signal utile. On choisit ici les hypothèses suivantes :

- Le densité spectrale du bruit est supposée constante sur toutes les fréquences. Son niveau dépend du rapport signal sur bruit *SNR*.

- Le signal est modélisé par un type de processus aléatoire appelé autorégressif d'ordre 1 (*AR(1)*), dépendant d'un paramètre *b*.

La fonction *wiener* dépend donc des deux paramètres suivants :

- *SNR* : Rapport de la puissance du signal utile (après avoir enlevé la valeur moyenne) sur la puissance du bruit. C'est nécessairement un réel positif. Plus le *SNR* est faible, plus le bruit est supposé important et donc plus le lissage sera fort (sur toutes les fréquences de façon uniforme).

- *b* : paramètre du modèle AR(1) du signal utile. *Il doit être compris entre 0 et 1 strictement*. Plus il est élevé, plus on considère que le signal est composé uniquement de faibles fréquences, et donc plus les variations rapides seront atténuées.

Ces paramètres peuvent être modifiés dans l'appel à la fonction *wiener*, dans la fonction *prefilter_part* du fichier *csv_processing.py*. Le deuxième argument est le *SNR* et le troisième est *b*.

2.3.2 Saturation en oxygène : filtrage avec un modèle Markovien

Pour la saturation en oxygène, on filtre le signal à l'aide de la fonction *markov*. Cette fonction renvoie un signal filtré, obtenu par la minimisation d'une fonction de coùt. Celle-ci contient un terme d'attache aux données qui oblige le signal filtré à être *proche* du signal original, et un terme de régularisation qui pénalise les différences entre échantillons consécutifs. L'importance de ce second terme est déterminé par un paramètre *beta* positif.

Ainsi, l'importance du filtrage peut être modifié en jouant sur le paramètre *beta* de la fonction *markov*. Si $beta = 0$, alors le signal n'est pas modifié du tout, et plus on augmente *beta*, plus le lissage est important.

Ce paramètre peut être modifié dans l'appel à la fonction *markov*, dans la fonction *prefilter_part* du fichier *csv_processing.py*. Le deuxième argument est *beta*.

2.4 Traitement des données de mouvement

Contrairement au rythme cardiaque et à la saturation en oxygène, les données d'accélération et de rotation peuvent évoluer beaucoup plus rapidement en cas de mouvements du patient. Il n'est donc pas souhaitable de

lisser les évolutions rapides de ces signaux, car en cas de mouvement, elles peuvent être très différentes entre deux échantillons consécutifs.

Les informations d'accélération et de rotation vont être combinées en une seule information représentant le mouvement du bracelet.

Pour cela on retranche d'abord à chaque type de données la médiane de ses valeurs. Dans le cas de l'accélération, cela doit être fait avec une fenêtre glissante pour prendre en compte le fait que cette valeur dépend légèrement de l'orientation du bracelet, et donc elle peut changer après chaque mouvement.

Puis ces nouvelles suites de valeurs sont chacune divisées par un certain quantile de leurs distribution de valeurs, de façon à normaliser leur variations et les rendre indépendantes des réglages des capteurs physiques.

On obtient alors une suite de vecteur de dimension 2 qui contiennent à la fois la donnée issue de la rotation, et celle issue de l'accélération. On les combine en calculant la norme euclidienne de ces vecteurs.

La donnée de mouvement ainsi obtenue présente de très fortes variations d'amplitude entre les phases d'éveil et de repos, ce qui rend difficile la lisibilité lors des phases de sommeil. Par conséquent, on applique une transformation logarithmique ainsi qu'un seuillage, ce qui permet d'atténuer les valeurs les plus fortes.

La donnée finale de mouvement est donc comprise entre 0 et 1, les valeurs à 1 correspondent aux forts mouvements.

2.5 Détection d'événements sur les signaux

Les étapes précédentes permettent d'obtenir des données préfiltrées puis filtrées de rythme cardiaque et de saturation en oxygène, ainsi que les données de mouvements. À cela s'ajoute le niveau d'intensité sonore, dont les données sont assez hétérogènes entre les différents fichiers.

Cette étape consiste à détecter les événements sur chacune de ces données, de façons à créer des nouveaux signaux entre 0 et 1 correspondant à la détection d'événements pour chacune d'elles.

Pour détecter les événements sur un signal, on calcule pour chaque instant l'écart-type des valeurs prises par le signal dans une fenêtre centrée sur cet instant. À partir de la distribution de ces valeurs d'écart-type, on choisit automatiquement un seuil et un facteur d'échelle. Ceux-ci vont être utilisées pour transformer ces écarts-types, à l'aide d'une fonction sigmoïde, en des valeurs comprises entre 0 (pas d'événements significatifs) et 1 (événement important sur le signal).

Cette détection est appliquée indépendamment sur les données préfiltrées de ryhtme cardiaque et de saturation en oxygène, ainsi que sur les données

de mouvement et de volume sonore.

On déduit alors de ces différentes détections une donnée de détection d'événement global. Pour cela, on multiplie les détections d'événement associées à chaque donnée (en inversant la détection uniquement le cas du volume sonore car c'est l'absence d'événement sur le signal qui peut être la marque d'un arrêt de respiration).

Cette détection globale permet de pondérer l'importance du filtrage des données de rythme cardiaque et de saturation en oxygène dans le réslutat final.

En effet, les valeurs écrites dans le csv de sortie sont des moyennes pondérées des données préfiltrées (uniquement l'interpolation linéaire pour les données manquantes ou aberrantes) et des données filtrées (par méthode de Wiener ou de modèle markovien). Pour chaque instant, ces deux données sont combinées, la valeur de détection globale d'événement est la pondération du signal préfiltré, et le complémentaire par rapport à 1 de cette valeur est la pondération pour le signal filtré. De cette façon, lorsqu'un événement est détécté, on donne moins d'importance au filtrage pour éviter de perdre des informations à un moment potentiellement important.

Cette option peut être contrôlée à l'aide de l'argument *event_detection* de la fonction *process_csv* ou *process_folder*. Ainsi, si *event_detection=False*, les données écrites dans le csv seront exactement les données filtrées, sans les combiner avec les données préfiltrées.

3 Affichage des résultat

Le programme permet l'affichage de graphiques représentant les différents signaux. Pour ce faire, il suffit de rentrer l'argument *plot=True* dans la fonction *process_csv* ou *process_folder*. Cela va ouvrir pour chaque partie exploitable du signal, des fenêtres qui présenteront le traitement de chaque colonne. Ainsi, le nombre de fenêtres peut être assez important si le fichier csv est long.

Ces résultats sont illustrés dans les figures suivantes. Les figures 1 et 2 présentent respectivement les données de rythmes cardiaque et de saturation en oxygène. Les courbes présentes sont celles des données brutes (en gris), des données préfiltrées par seuillage et interpolation linéaire (en vert), puis des données traitées par filtrage de Wiener ou méthodes markoviennes (en bleu), et enfin le mélange de ces deux dernières en prenant en compte la détection d'événement (en rouge).

Les données d'intensité sonore et de mouvement correspondent aux figures 3 et 4. Les données brutes sont représentées en gris, et la détection d'événement (ou de silence pour le cas de l'audio) est représentée en jaune.

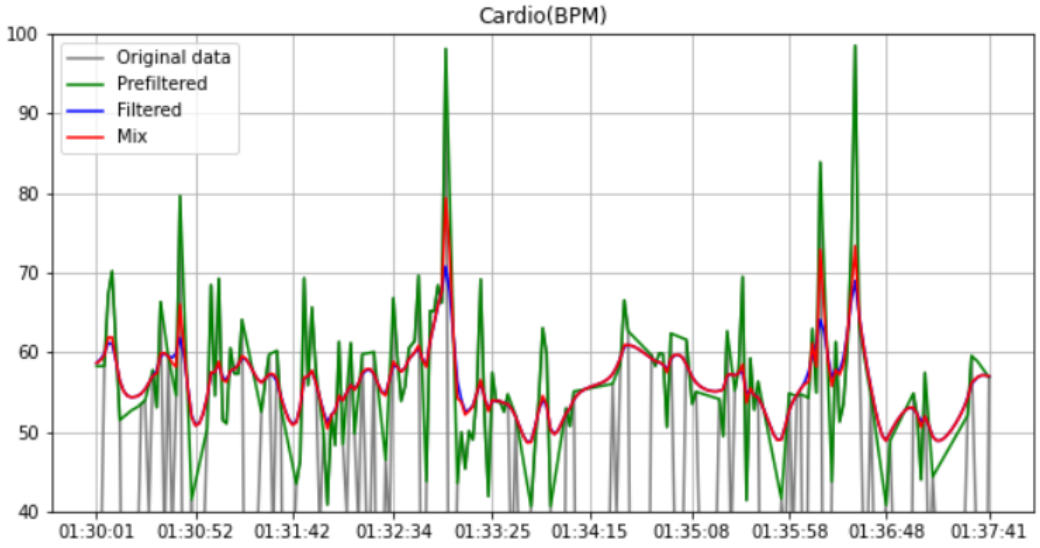

FIGURE 1 – Rythme cardiaque

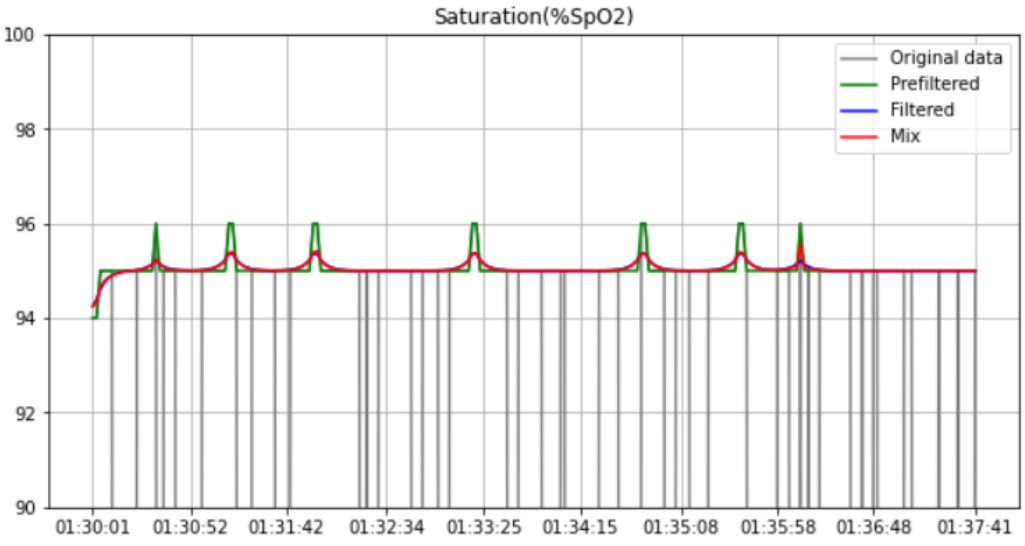

FIGURE 2 – Saturation en oxygène

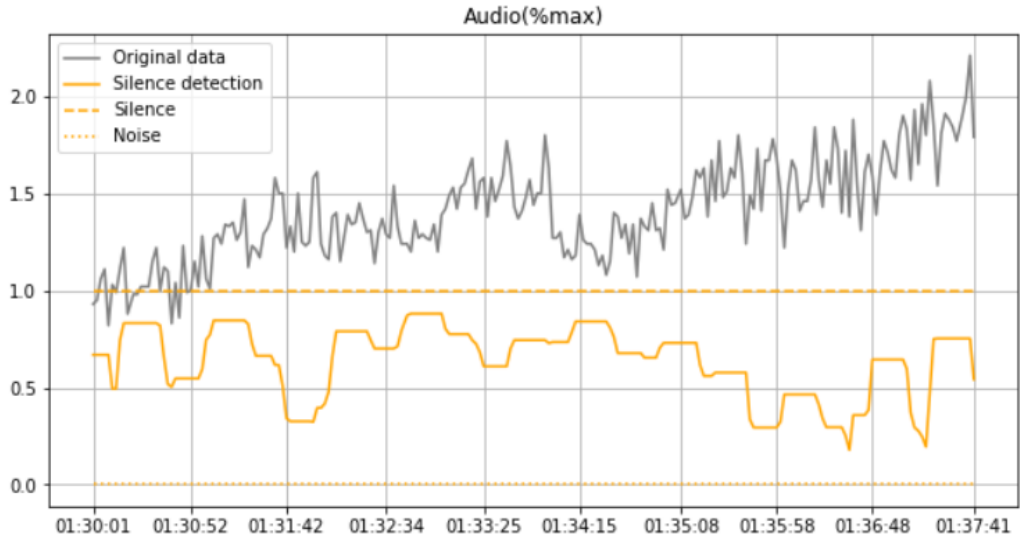

FIGURE 3 – Intensité sonore et détection de silence

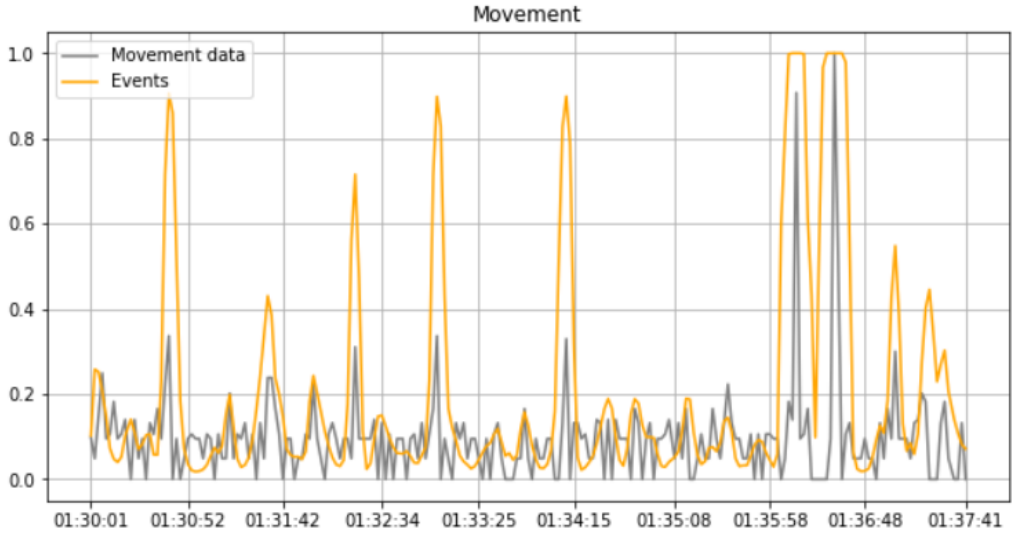

FIGURE 4 – Mouvement et détection d'événements

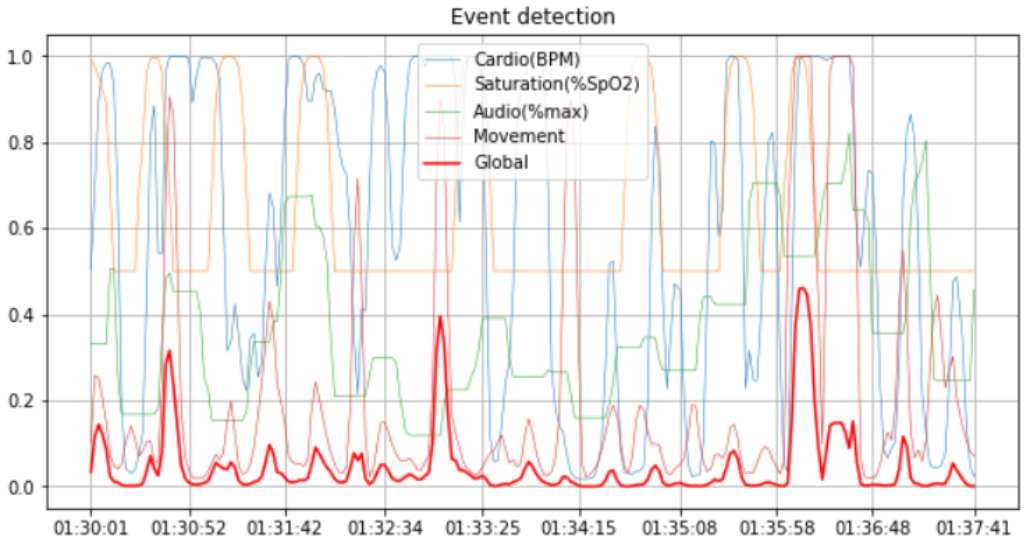

FIGURE 5 – Détection d'événements

La dernière figure (5) montre la détection d'événement sur les différentes données. La courbe rouge correspond au produit de toutes les autres (en ayant changé le sens pour l'audio comme expliqué en 2.5).