Rapport de Projet

Stylo intelligent

Carrand Adrien
Brissonneau Nicolas

PJE09 2016

Table des matières

BRISSONNEAU Nicolas
CARRAND Adrien

BRISSONNEAU Nicolas
CARRAND Adrien

Remerciements

Nous tenons à remercier M. Monteiro pour nous avoir proposé ce sujet ainsi que pour son encadrement. Nous souhaitons également remercier M Rébillat pour son aide précieuse ainsi que la rapidité avec laquelle il nous a fourni du matériel ; enfin M Guskov pour le temps qu'il nous a accordé.

Introduction et Contexte

Ce projet rentre dans le cadre des projets PJE09 en troisième année des Arts et Métiers. Il a pour thématique le stylo intelligent, c'est-à-dire un stylo avec lequel ce que l'on écrit se traduit en texte exploitable par un logiciel de traitement de texte. Il s'agit donc d'un système permettant l'acquisition d'écriture manuscrite et sa numérisation. Pour cela, nous détaillerons notre démarche et nos choix :

- Technologie employée,
- Chaine d'acquisition,
- Traitement des données,
- Reconnaissance de formes algorithmique.

Notre cahier des charges sera le suivant, il nous aiguillera sur le choix des technologies et des algorithmes employés :

- Simplicité d'utilisation : « Plug & Play »,
- Réactivité du système : de l'ordre de quelques secondes,
- Encombrement : minimum,
- Temps de réalisation du projet : celui d'un PJE09,
- Reconnaissance : forme simples

En ce qui concerne la reconnaissance de forme, nous partons sur l'idée que nous n'allons reconnaitre dans un premier temps que des formes simples. Si le système se comporte bien, nous complexifierons l'épreuve du système, avec des chiffres, puis éventuellement des lettres.

Au sujet du temps de réalisation du projet, cela signifie que nous ne pouvons pas nous lancer dans de la recherche pure, en effet nous consacrons à notre projet une journée complète par semaine. Nous devons donc être efficace, tout en concevant la visée pédagogique de ce projet.

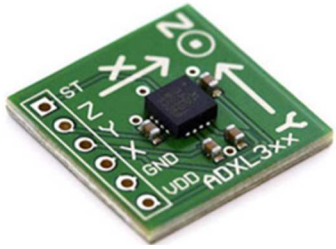
Choix technologiques & stratégie

État de l'art des technologies disponibles

Le stylo permettant la capture d'écriture manuscrite a été étudié, puis industrialisé, par de nombreuses entreprises et laboratoires de recherche, citons notamment l'entreprise suédoise Anoto. Ainsi, plusieurs solutions technologiques s'offrent à nous :

- L'accéléromètre :

De petite taille, et simple d'utilisation, l'accéléromètre permettrait un prototypage simple et efficace. De plus, il existe une littérature scientifique accessible concernant l'exploitation d'un tel capteur à des fins de reconnaissance de forme.



- La technologie dite « active » :

Cette technologie implique l'emploie d'une surface adaptée, comme une tablette par exemple. Elle permettrait de résoudre simplement le problème de localisation du stylo par exemple. Néanmoins elle est couteuse et se révèle assez encombrante.

BRISSONNEAU Nicolas
CARRAND Adrien

- La technologie à capteur de position déporté :

Cette solution est moins chère que la précédente, mais représente tout de même une contrainte pour son utilisateur, celle de fixer le récepteur et de signifier chaque changement de page. Sur l'illustration, il s'agit d'un stylo à recepteur infrarouge.

- Méthode impliquant une caméra :

Sans doute la solution la plus intuitive : une caméra placée au bout du stylo, près de la bille, suit les mouvements du stylo en observant les mouvements relatifs par rapport à la feuille. Cette solution est développée par Anoto. Mais cette méthode reste couteuse, la caméra nécessaire (bien qu'une résolution élevée ne soit pas utile) est chère puisque fortement miniaturisée. De plus, un papier particulier est nécessaire, en effet ce papier affiche des points de repère qui permettent à la caméra de comprendre le mouvement.

- Le « Trackball Pen » :

Il s'agit ici d'un stylo dont la bille est reliée à des capteurs permettant de connaitre ses rotations, et donc par extension de connaitre son tracé. Cette technologie est la plus opaque du point de vu de la littérature, puisque nous n'avons pratiquement rien trouvé concernant cette solution. La développer nous-même nous semble fort complexe, ainsi que couteux en temps et en argent.

Etant donnés ces possibilités, nous allons faire notre choix au regard de notre cahier des charges.

Liberté de mouvement

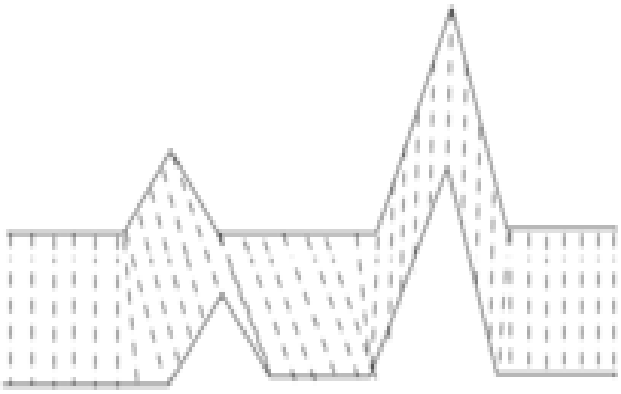
Nous avons choisi de développer notre technologie à partir d'un accéléromètre, que nous exploiterons à l'aide d'une carte type Arduino et du logiciel de calcul Matlab. En effet ces systèmes d'acquisition et de traitement de donnés sont abordables et nous savons déjà nous en servir.

Cette solution se justifie d'une part par le type d'application qu'elle propose. En effet, un accéléromètre permet d'élargir le champ d'application du stylo à la 3D, tout en restant une technologie embarquée, ce qui aurait été un réel challenge avec les autres solutions (ainsi nous répondons notamment à la contrainte d'encombrement).

D'autre part, il s'agit de la seule solution qui demande essentiellement un travail de traitement de données. Ainsi, dans le cadre de ce projet, nous avons favorisé la solution qui nous permettait de mettre en place un système viable en peu de temps afin de respecter les délais tout en mettant en avant notre formation en mécatronique.

Concernant la reconnaissance de forme, nous choisissons l'algorithme du « Dynamic Time Warping » (ou Déformation Temporelle Dynamique), dit DTW dans la suite de notre rapport. Le Dynamic Time Warping est une méthode d'analyse et de quantification des similarités entre deux signaux distordus en amplitude comme en temporalité.

Il est particulièrement connu pour avoir prouvé son efficacité dans des problématiques de reconnaissance de la parole, où la vitesse de prononciation et l'amplitude de la voix varient d'une personne à l'autre. En effet, il est le plus adéquat à notre problème : de la même manière qu'il a permis d'identifier la parole malgré la signature vocale de chacun, dans notre étude nous aurons à comparer des signaux qui seront les accélérations du stylo lorsqu'il est utilisé pour écrire. Étant donné qu'il s'agit d'un mouvement produit par un être humain, il ne sera pas exécuté deux fois avec la même amplitude ni à la même vitesse.



Valider la faisabilité du projet

Notre démarche sera la suivante : tester une première fois l'ensemble de la chaine d'acquisition et de traitement de données afin de vérifier la faisabilité du projet avec nos choix. Puis, dans un deuxième temps, et cela représentera la majorité du temps investi dans notre projet, nous affinerons chacune des fonctions utilisées dans ce projet afin de donner du sens et d'optimiser nos résultats. Ainsi, nous concentrons nos efforts sur l'optimisation de la chaine d'acquisition ainsi que le post-traitement d'un côté, et sur l'efficacité de l'algorithme de reconnaissance de forme d'un autre.

Chaine d'acquisition

Afin d'exploiter les données de l'accéléromètre, nous avions besoin de mettre en place une chaine d'acquisition. L'architecture choisie sera la suivant : Accéléromètre, puis un microcontrôleur (type Arduino), puis un ordinateur avec Matlab qui traitera les données et les comparera à l'aide du DTW, enfin une passerelle entre MATLAB et Word permettra le traitement final.

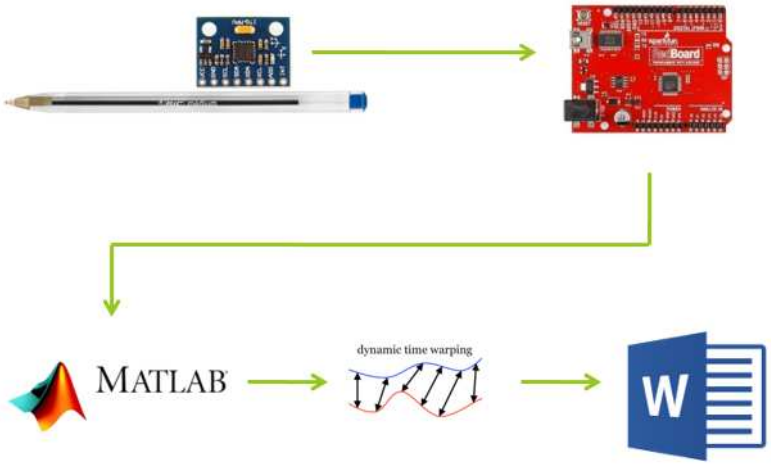
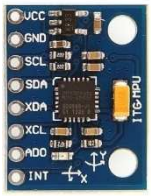


Schéma de principe du prototype

BRISSONNEAU Nicolas
CARRAND Adrien

Capteur



Nous avons choisi l'accéléromètre MPU6050 pour son prix faible, sa simplicité d'utilisation, et sa disponibilité immédiate. A cet égard nous remercions M. Marc Rébillat pour nous en avoir fourni un gratuitement. Ce capteur nous fournis l'accélération suivant les trois axes de son repère orthonormé, ainsi que les vitesses angulaires autour de ces derniers.

Interface Arduino

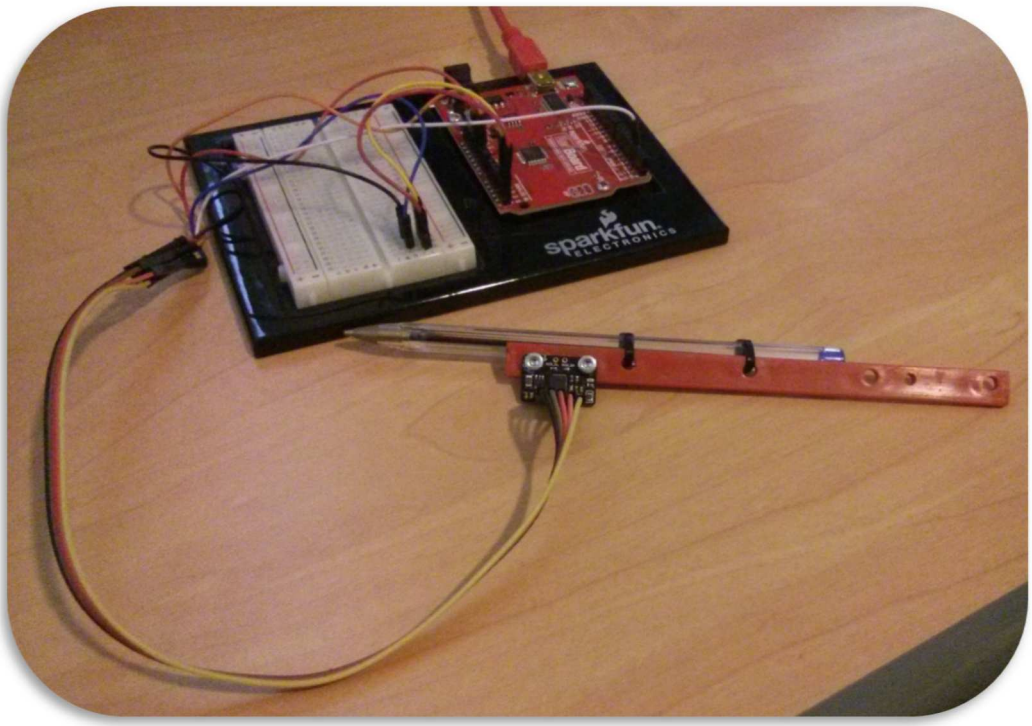
Nous avons choisi la carte RedBoard de SparkFun, qui est une copie légale de la fameuse carte Arduino. Nous remercions M. Éric Monteiro pour nous en avoir fourni une très rapidement. Sa simplicité d'utilisation et sa très large communauté (celle d'Arduino en général) nous ont aidé à avancer rapidement.

Matlab

Nous choisissons d'utiliser le logiciel Matlab. En effet, il est au cœur de l'enseignement du notre semestre d'expertise, nous commençons donc à être plutôt à l'aise avec. En outre, il est parfaitement à même de réaliser sa tâche, qui consistera principalement à la gestion de données par tableau et à appliquer des fonctions mathématiques demandant une certaine puissance de calcul.

Assemblage du prototype :

Une fois tous ces éléments reçu et assemblés, notre prototype est prêt à être utilisé. Ci-dessous une photographie de notre prototype :



BRISSONNEAU Nicolas
CARRAND Adrien

Prétraitement des données

Nous réalisons donc l'acquisition des données sortant de notre accéléromètre. Dans un premier temps, nous exécutons un code Arduino, à travers la plateforme de développement Arduino, pour vérifier que nous en sommes capable. Ce code est disponible en annexe sous le nom de *MEMS-9DOF-VectorData.ino*. A l'aide de la console de cette plateforme, nous pouvons voir défiler des valeurs codées sur 16bit, donc à un coefficient près nous avons les valeurs d'accélération et de vitesses angulaire sur chaque axe.

Avant toute chose il a donc fallu déterminé ces coefficients, ainsi que le biais de chaque variable, afin d'étalonner le capteur. En effet elles n'étaient pas tout à fait centrées en zéro. Pour les accélérations, nous savons que ce capteur mesure à chaque instant, en plus des accélérations dans le repère terrestre, l'accélération de la pesanteur.

En considérant qu'elle est égale à g=9.81m/s², nous avons simplement mis le capteur dans une position telle que la pesanteur soit ressentie entièrement par un axe, par exemple Z. Nous avons fait une acquisition de 100 valeurs de sortie du capteur sur l'axe Z à l'aide de Matlab, en effet le signal est bien entendu bruité. Nous avons ensuite moyenné cette valeur, ce qui nous a donné « Zgplus ». Puis nous avons fait la même chose mais en retournant le capteur de manière à ce que la pesanteur soit entièrement ressentie sur -Z. Nous avons acquis 100 valeurs que nous avons moyennées, obtenant « Zgmoins ». Ainsi, nous formions les valeurs suivantes :

$$SF_Z = \frac{Zgplus - Zgmoins}{2 * g}$$

$$B_Z = \frac{Zgplus + Zgmoins}{2}$$

Nous avons réalisé les mêmes manipulations pour obtenir SF_X, SF_Y, B_X, B_Y. Ainsi, nous pouvions corriger les biais et erreur d'échelle par la formule suivante :

$$X_{mesuré} = \frac{X_{capteur} - B_X}{SF_X}$$

$$Y_{mesuré} = \frac{Y_{capteur} - B_Y}{SF_Y}$$

$$Z_{mesuré} = \frac{Z_{capteur} - B_Z}{SF_Z}$$

Nous n'avons eu réellement besoin des gyroscopes qu'assez tard dans le projet, donc ce qui suit est ben plus loin chronologiquement dans le projet. Pour étalonner les gyroscopes, qui mesurent la vitesse angulaire, nous avons procédé de manière similaire. Pour récupérer leurs biais, nous avons fait une mesure des trois variables avec le capteur immobile. En effet, s'il n'y a pas de mouvement, les gyroscopes sont censés être à zéro. En prenant 100 acquisitions, puis en les moyennant, nous avions directement les biais $Bgyro_X$, $Bgyro_Y$, et $Bgyro_Z$.

Comme l'indique le code *MEMS-9DOF-VectorData.ino*, la carte Arduino nous fournit le temps écoulé entre chaque mesure. Nous avons donc pu remarquer que ce temps entre chaque retour d'information n'est pas constant, bien que globalement autour de 38ms. Nous rajoutons donc une condition dans le code pour que les données soient lues et envoyées à Matlab toutes les 41ms. Ainsi nous avions une base de temps. Cette amélioration est arrivée en milieu de projet environ.

Nous faisions une acquisition d'un mouvement particulier : nous tenions le capteur monté sur le stylo de manière immobile en début d'expérience, puis nous le tournions d'un demi-tour autour de l'axe que nous souhaitions étalonner. Ainsi, il suffisait d'intégrer ces valeurs, c'est-à-dire les multiplier par le delta-t et les sommer, pour obtenir une valeur, Gx par exemple. Nous avions donc la relation suivante :

$$SF_{gyroX} = \frac{Gx.\Delta t}{\pi}$$

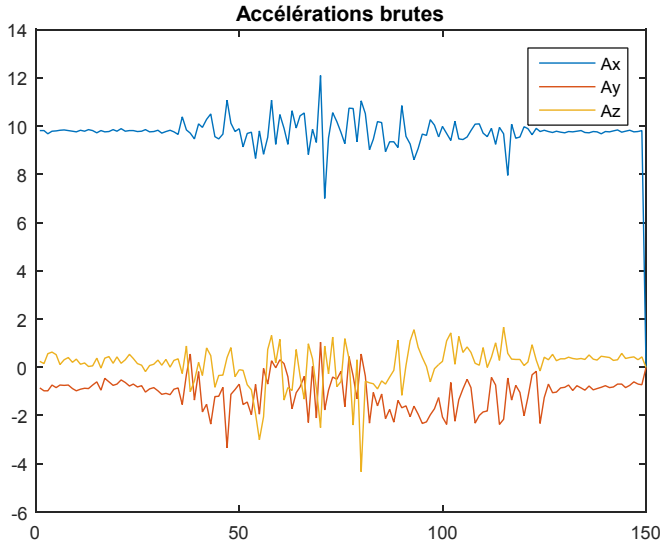
Nous obtenons donc des relations similaires aux accélérations :

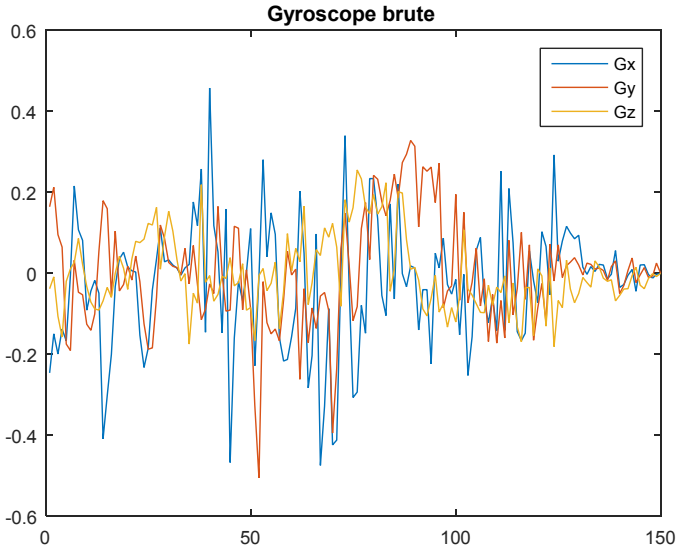
$$\omega X_{mesuré} = \frac{\omega X_{capteur} - Bgyro_X}{SF_{gyroX}}$$

$$\omega Y_{mesuré} = \frac{\omega Y_{capteur} - Bgyro_Y}{SF_{gyroY}}$$

$$\omega Z_{mesuré} = \frac{\omega Z_{capteur} - Bgyro_Z}{SF_{ZgyroY}}$$

Une fois ce calibrage effectué, on peut effectuer des acquisitions. Ci-dessous, les accélérations et vitesses angulaires pour un cercle écrit avec notre stylo :





BRISSONNEAU Nicolas
CARRAND Adrien

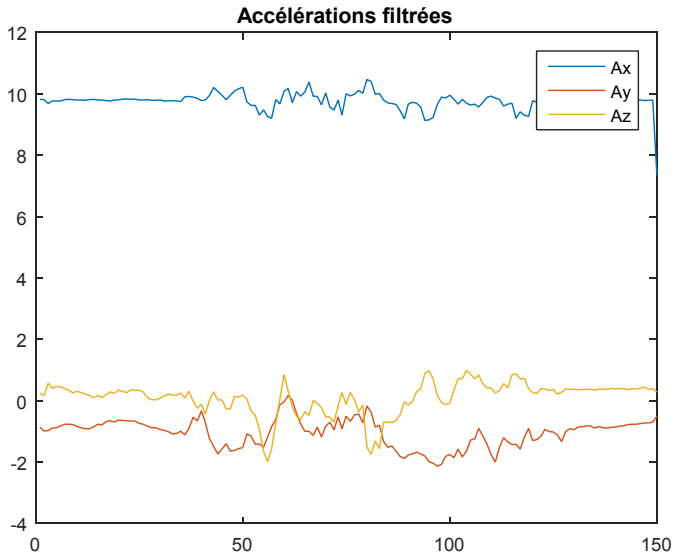
Filtre passe bas

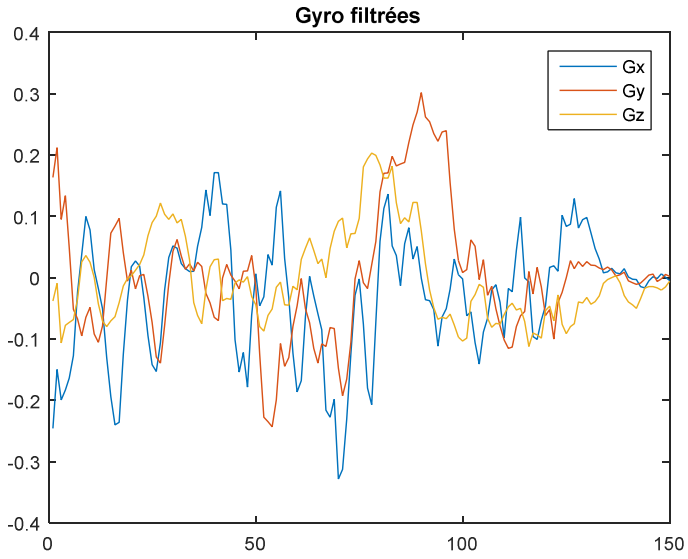
Comme nous pouvons le remarquer, ces valeurs sont entachées de bruit. Nous supposerons pour le reste de ce rapport qu'il s'agit de bruit blanc de moyenne nulle. Nous allons donc appliquer un filtre passe bas sur ces acquisitions. Nous choisissons de mettre en place une moyenne glissante, qui fera en sorte que chaque valeur filtrée est la moyenne d'un nombre paramétrable de valeurs brutes. Cette moyenne glissante s'exécute par le code Matlab suivant :

```
Vtemp=Data

for j=0:N-Te
    Vtemp(N-j,:)=(sum(Vtemp((N-j)-(Te-1):N-j,:)))/Te;
end
```

Avec Data le tableau recueillant les acquisitions. Chacune de ses lignes est une acquisition à un instant donnée, et les colonnes correspondent aux variables dans l'ordre suivant : AccX, AccY, AccZ, GyroX, GyroY, GyroZ. Le résultat de Data moyenné avec un pas de 4 donne les courbes suivantes :





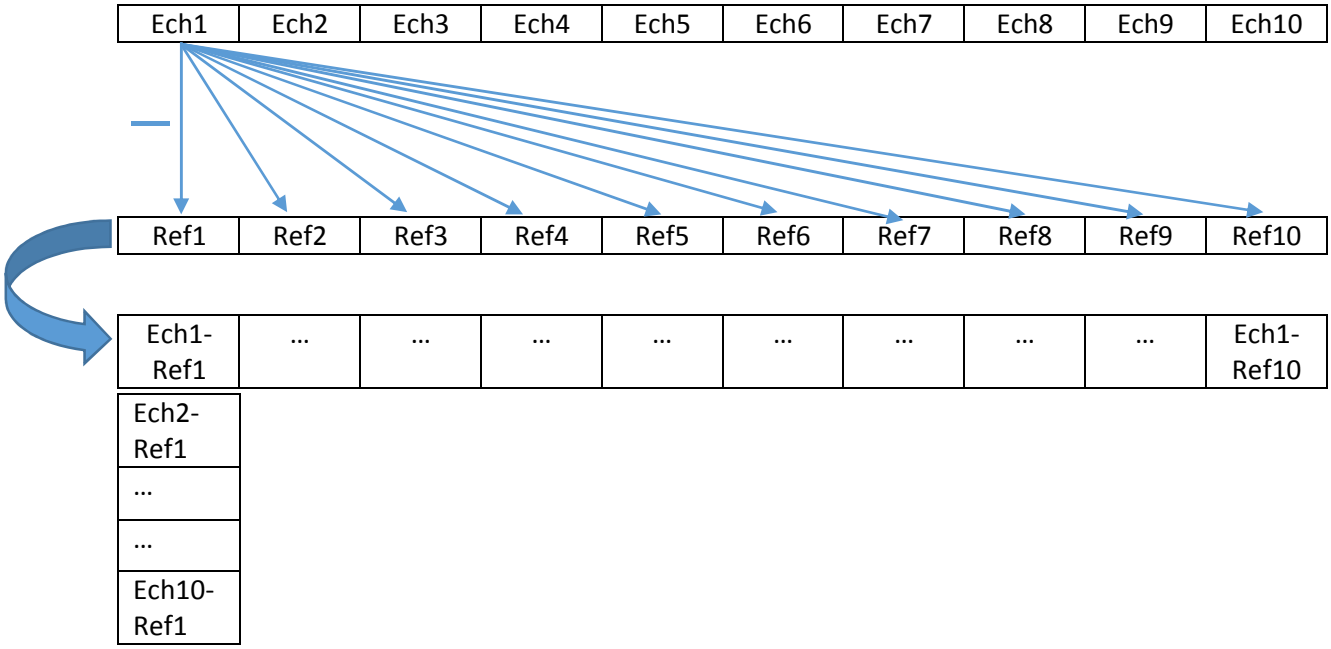
Par la suite, nous avons testé différentes valeurs pour le nombre de pas de la moyenne glissante. Après plusieurs essais/erreurs, nous concluons que 4 est la valeur qui nous convient le mieux.

Mise en place de classe pour la reconnaissance de similitude

Comment allons-nous définir une classe ? Ou plutôt, comment l'algorithme de DTW va-t-il se servir de celle-ci ? En réalité, il va la considérer de la même manière qu'il considère les échantillons à tester, ainsi il ne répond que partiellement aux attentes que l'on a de la classification. L'algorithme ne faisant pas de distinction sur ce point-là, nous allons purement observer une quantification de la similitude entre les deux signaux. Pour ce faire, pour un échantillon d'accélérations en X de dimension $nx1$ associé à une classe en X de dimension $nx1$, un tableau 2d nxn sera construit, basé sur la différence entre chacune des n accélérations de l'échantillon test avec chacune des n valeurs issues de l'échantillon de référence.

BRISSONNEAU Nicolas
CARRAND Adrien

Prenons un exemple à faible dimension pour clarifier le sujet :



Chacune des cases de ce tableau de dimension nxn contiendra un coût, soit une quantification de l'écart en amplitude entre une valeur de l'échantillon et une valeur de la référence. En considérant les extrémités de la diagonale $(1\ ;\ 1)$ et $(n\ ;\ n)$ comme étant respectivement les points de départs et d'arrivée, il faudra alors tracer un chemin entre ces deux cases en sommant les coûts des cases empruntées le long du chemin: l'objectif de l'algorithme sera alors de trouver le chemin le moins « coûteux ». Ainsi la distorsion en amplitude est évaluée par le coût et la distorsion temporelle est repérée par l'écart entre le chemin calculé et celui de la diagonale.

En remarquant que nous effectuons une quantification de similarité sur des grandeurs d'accélération linéaires d'une part et de vitesses angulaires d'autre part, on comprend qu'il peut y avoir conflit, prépondérance d'une grandeur physique sur une autre. C'est pourquoi nous décidons de normaliser nos mesures en fixant préalablement des bornes saturantes min-max trouvées empiriquement, nos valeurs seront donc toutes comprises entre -1 et 1 avec une pondération presque équivalente entre les différentes dimensions.

Fonctionnement continu

Afin de mieux comprendre le comportement des variables d'accélérations et de vitesses angulaires, nous avons mis en place un code mettant à jour les graphes à chaque instant d'acquisition, ce qui nous donne un rendu de fonctionnement continu. Ainsi nous pouvions voir en temps réel nos acquisitions, nous assurant ainsi que ce qu'il se passait à l'écran correspondait bien à la réalité.

Nous avons ensuite appliqué le code de moyenne glissante aux valeurs acquises, ainsi nous pouvions nous assurer du comportement de notre filtre, et des données qui entrait dans notre algorithme de détection de similitude.

Valeur seuil définissant le mouvement

Une autre problématique qui se posait à nous était de déterminer à quelle moment une acquisition était intéressante et méritait d'être traité. En effet, pour que le DTW fonctionne correctement, il faut que le signal qui est comparé démarre de la même manière que le signal de référence. Ainsi, si nous avions dessiné un cercle en commençant par le haut, et que le « cercle de référence » avait été tracé en commençant par le bas, notre algorithme nous aurait retourné un indice élevé, c'est-à-dire une faible similitude.

Nous avons donc travaillé sur la norme des accélérations mesurées, si celle-ci dépassait une valeur seuil. Étant donné que l'accélération de la pesanteur est toujours détectée, à l'arrêt la norme était égale à 9.81m/s². En mouvement, cette valeur pouvait être supérieur, ou inférieur (en cas de chute par exemple). Nous avons donc considéré la condition suivante sous Matlab :

```
if (norm(Vect(i,1:3))-9.81)<0.1
```

Étant donné le bruit blanc que nous avions toujours un peu puisque nous ne voulions pas que le filtre détériore trop le signal, il s'est avéré expérimentalement que la valeur de 0.1m/s² comme valeur seuil fonctionnait bien.

Une fois cette valeur seuil franchie, notre programme considérait que le stylo était en mouvement. A ce moment-là, une acquisition se construisait, jusqu'à ce que la norme de l'accélération retombe à 9.81. Nous considérons cette acquisition comme une « acquisition de mouvement ». Nous considérions que les mouvements étaient relativement rapides, c'est-à-dire, étant donné la dynamique du système, que le système n'a pas le temps de se stabiliser durant une phase de mouvement.

Cette méthode nous permet de découper le flux d'acquisition selon si le stylo est en mouvement ou non. Ainsi, il facilite l'emploi de l'algorithme du DTW.

Sauvegarde des données (permettre l'analyse des résultats)

Nous mettons également en place un code permettant de sauvegarder dans des fichiers .csv les acquisitions du stylo en mouvement, en vue d'un post-traitement ultérieur, en dehors du programme principal. Cette méthode nous permettra de débugger le code principal ainsi que ses fonctions. En effet, en fonctionnement continu, nous n'avons jamais vraiment deux fois la même acquisition, ce qui peut être très problématique dans des phases de débugge, qui sont inévitables.

Premiers tests complets

Nous avions donc tous les éléments pour procéder à un premier test de l'ensemble de notre système. Nous avons créé deux classes : une classe cercle, et une classe carré. Pour former ces classes, nous avions besoin d'un échantillon « type » des patterns d'accélérations. Pour obtenir cet échantillon type, nous avons effectué plusieurs acquisitions des accélérations générées lors du mouvement décrivant un cercle. Il s'agissait toujours du même cercle, parcouru de manière régulière, et à la même vitesse. Une seule et même personne devait effectuer ces cercles à l'aide du stylo, car chaque personne à sa propre manière de déplacer le stylo. Une fois toutes ces acquisitions enregistrées, nous en avons fait la moyenne, en faisant attention à ce qu'elles démarrent bien toutes au même moment et finissent toutes également à un même instant. Cela impliquait des temps d'acquisition égaux.

Nous avons donc mis cet échantillon moyenné comme référence de notre classe cercle. Pour vérifier que notre code de DTW fonctionnait, nous avons lancé des acquisitions en fonctionnement continu, comme décrit précédemment. Nous mettions notre stylo en position, à l'arrêt, puis nous dessinions un cercle de la même manière que pour la création de classe. Ce test était concluant : les valeurs de sorties de la fonction étaient relativement faibles, entre 0.8 et 0.2, c'est-à-dire qu'il détectait une bonne similitude entre nos mouvements et le signal de référence.

Fort de ce premier test, nous avons procédé de même pour le carré, que l'algorithme arrivait à reconnaitre avec une qualité légèrement inférieure, les valeurs de similitudes étaient comprises entre 0.3 et 1 selon les essais.

Puis nous avons voulu confronter les deux classes, c'est-à-dire lancer une acquisition en mode continu, et pour chaque acquisition de mouvement, un premier appel de la fonction DTW comparait ce signal au signal de référence du cercle, puis un deuxième appel de cette fonction au signal du carré. C'est alors que les problèmes commencèrent : que nous fassions des cercles ou des carrées, l'algorithme ne faisait pas la différence, et donnait de bons résultats pour à la fois le cercle et le carré. Nous sommes sûr du code du DTW, nous avons donc mis en doute nos acquisition, ce qui a soulevé une série de nouvelles problématiques.

Nouvelles problématiques

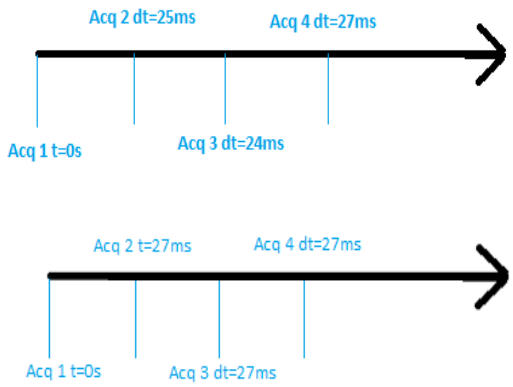
Alors que la chaîne d'acquisition s'est montrée fonctionnelle, le prétraitement des données s'est avéré comporter des lacunes. Auparavant, l'objectif du prétraitement était de recueillir des données cohérentes : faiblement bruitées, utiles et correctement enregistrées. Cependant, l'objectif principal du prétraitement est en réalité de préparer les données de manière à ce qu'elles soient exploitables par l'algorithme de DTW tout en conservant leur sens physique.

Base de temps

La première lacune considérée porte sur la base de temps. En effet, les informations reçues par l'accéléromètre sont délivrées à intervalles irréguliers. Si un avantage du DTW est qu'il nous permet de trouver une cohérence entre des signaux temporellement distordus, cette irrégularité reste une incohérence handicapante vis-à-vis de la qualité de comparaison des signaux.

En effet, si l'on se rappelle de la description du fonctionnement de l'algorithme de DTW, une base de temps dynamique ne peut que faire tendre le chemin parcouru à s'écarter de la diagonale.

Minimiser cet écart en fixant une base de temps d'acquisition sur l'acquisition des données nous permit de régler ce problème. Cependant la base de temps fixe impose l'inconvénient de diminuer la fréquence d'acquisition moyenne, elle consiste en la majoration du temps d'acquisition car on choisit la fréquence la plus lente pour s'assurer de ne pas perdre de valeurs.

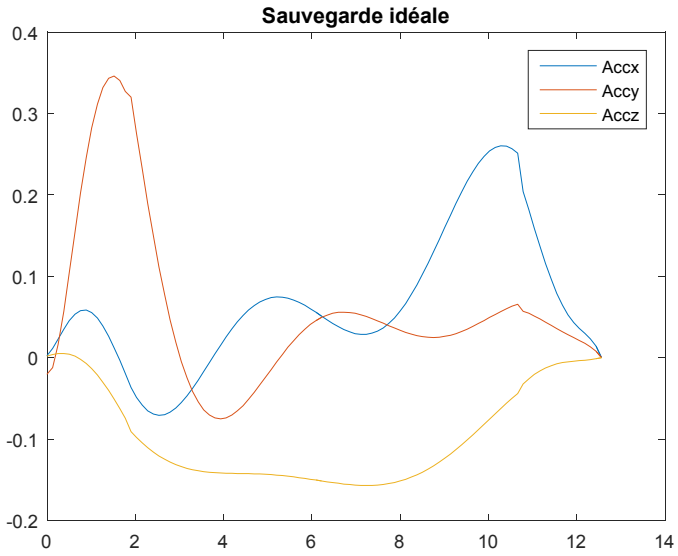


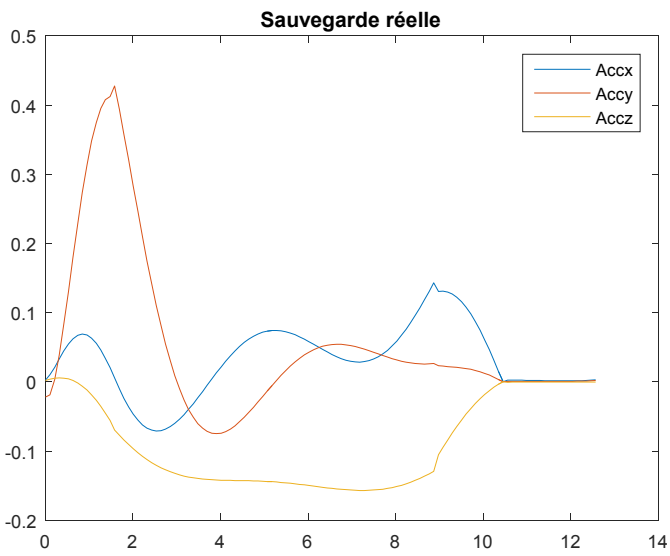
Définition de la base de temps fixe

Prétraitement des données pour le DTW

Précision de la détection de mouvement

Si la valeur seuil définissant la prise de mesure a été défini, ce dernier souffre pourtant d'un manque de précision. En effet, les résultats enregistrés montraient des défauts, soit à travers une quantité de données considérées utiles trop importante ou trop faible, ou encore en ne montrant parfois aucune donnée utile. L'ajustement de la détection de mouvement devait donc être améliorée. Pour cela, nous avons mis en place un système plus précis, en ajoutant une variable résultante de la dérivée de la norme pour signifier une variation importante d'accélération, alors signe de mouvement.

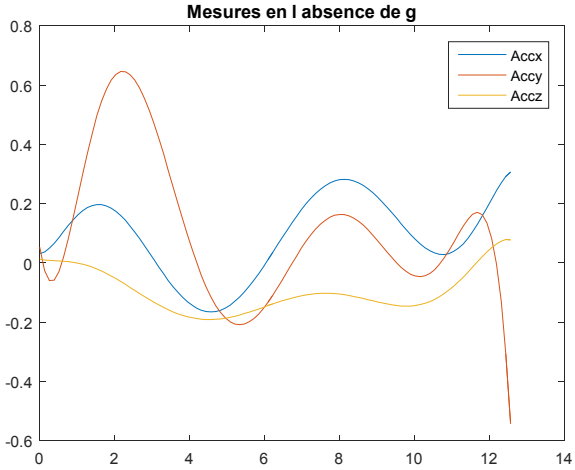


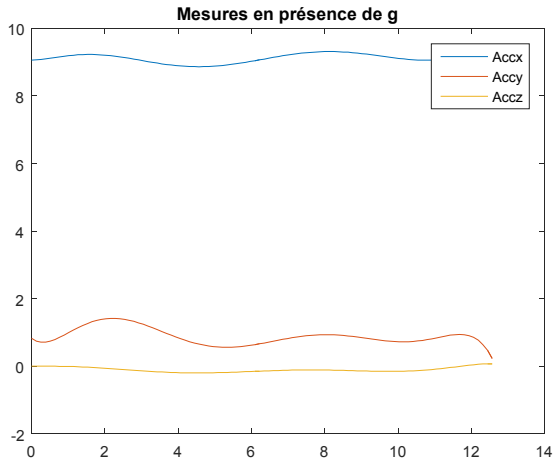


BRISSONNEAU Nicolas
CARRAND Adrien

Présence de g

L'élément le plus limitant dans la reconnaissance de signaux à ce stade du projet est l'omniprésence de la gravité dans nos mesures. Cette dernière, d'amplitude 9,81m/s², prend le pas sur nos mouvements dont l'amplitude en accélération varie autour de 0,5m/s² et elle finit par les « noyer ». On se retrouve alors dans la situation où l'on souhaite retrouver une allure d'accélération assimilable à la figure de gauche. Mais la présence de la gravité fait perdre aux courbes d'accélération les allures par lesquelles elles sont identifiables, comme montré ci-dessous à droite.



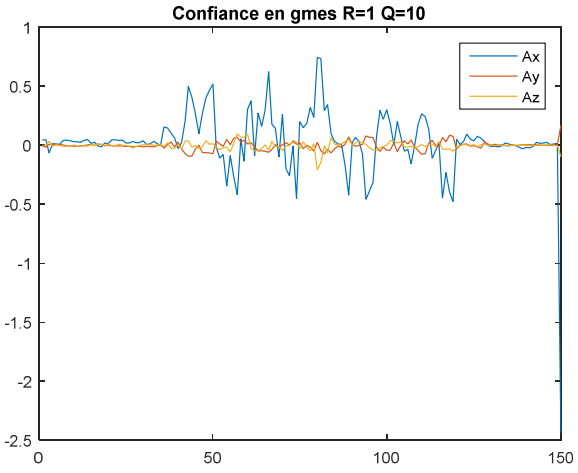


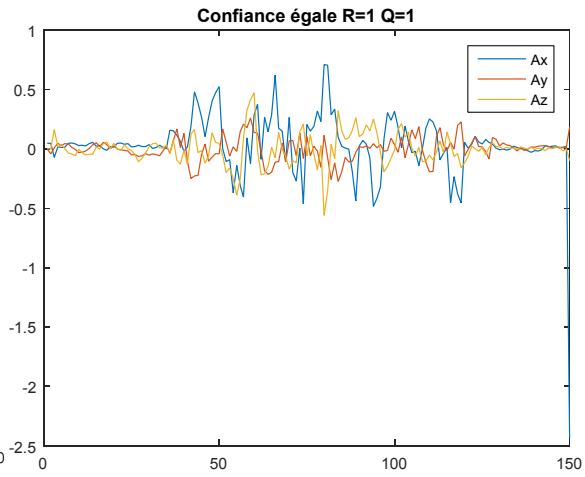
Nous avons donc cherché à développer une méthode pour retrancher g de nos acquisitions. En première approche, nous avons considéré que nos accélérations étant relativement petites devant g, la direction du vecteur 3D mesuré par notre capteur donnait la direction de de la pesanteur. Il suffisait donc de retrancher de manière pondéré 9.81 à ce vecteur pour retomber sur les accélérations pures :

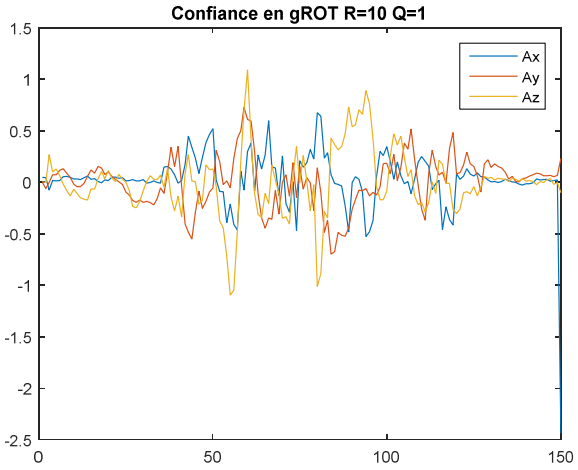
```
norme=norm(Vtemp(i,1:3));
gmes=((9.81/norme)*[Vtemp(i,1:3)])';
Vtemp(i,1:3)=Vtemps(i,1:3)-gmes;
```

Mais en réalité, nos accélérations sont parfois de l'ordre de g, notre première méthode fausse donc nos mesures. Nous avons donc eu une deuxième approche : corriger cet estimation du vecteur g à l'aide d'un filtre de Kalman. Le code de ce filtre est donné en annexe sous le nom de *KalmanG.m*.

L'idée de ce filtre était d'estimer la direction de g, en maintenant sa norme à 9.81m/s². A chaque pas de calcul, nous estimions la nouvelle direction de g à partir d'une matrice de rotation extraite des vitesses angulaire intégrée relevé durant le pas, la confiance dans cette méthode étant pondéré par la matrice Q. Ensuite, notre correction était extraite de la mesure des accélérations auxquelles nous retranchions g suivant la manière décrite en première approche, la confiance dans cette méthode est pondérée par la matrice R.







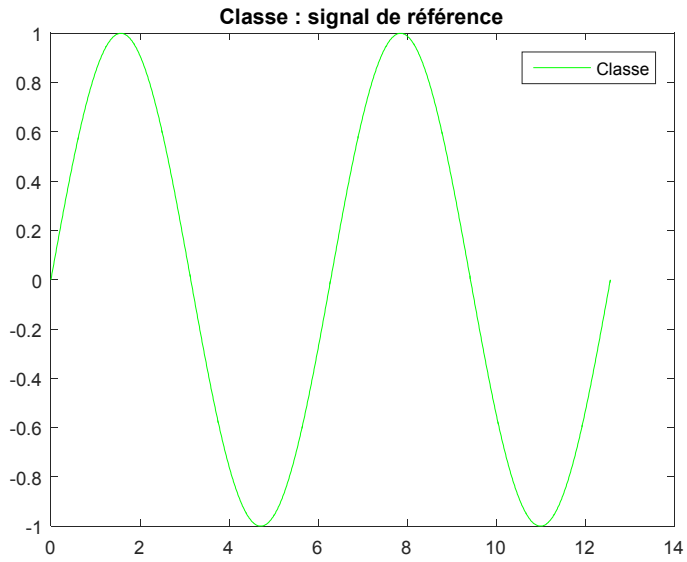
Voici ci-contre les courbes du signal présenté dans la partie *Prétraitement des données* traité avec notre filtre de Kalman sur g. Ce signal représentait un cercle tracé avec le stylo suivant l'axe des x.

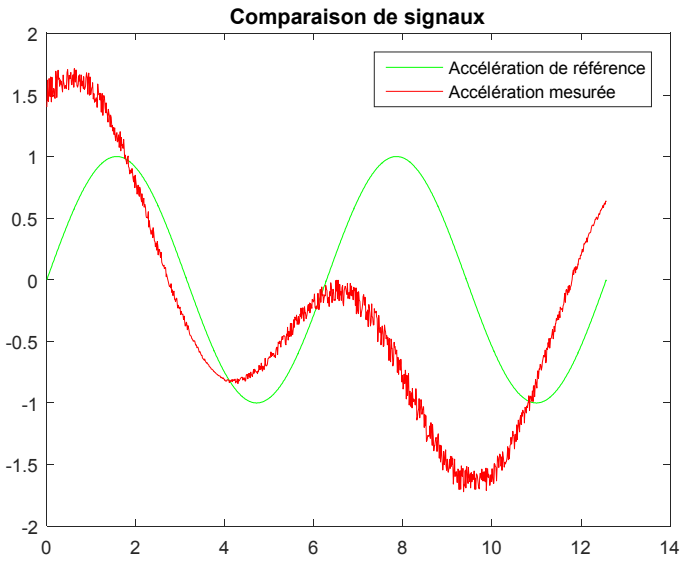
Ainsi, après plusieurs essais/erreurs, nous avons posé Q=10 et R=1, car nous avions peu confiance dans le gyroscope à ce moment-là. Nous avions l'impression qu'il faussait nos résultats.

Impacts sur le DTW

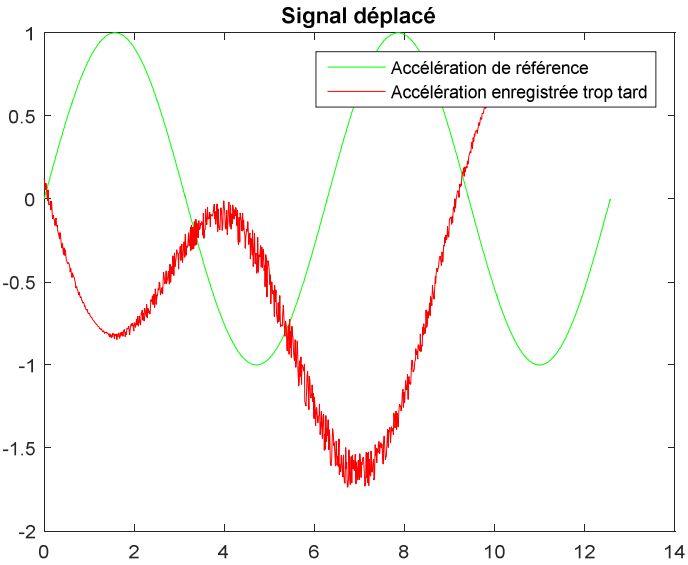
Lors de la mise en place de l'algorithme de DTW, nous avons fait l'hypothèse que les conditions de création de la classe étaient suffisantes pour aboutir à une reconnaissance de signal. Pourtant, cette dernière est dégradée par chacune des problématiques que nous avons listés précédemment. Ainsi, pour apporter une meilleure visibilité sur le problème, mettons en avant chacun des défauts décrits en supposant que l'on recherche une similitude avec une accélération sur x assimilable à la fonction sinus.

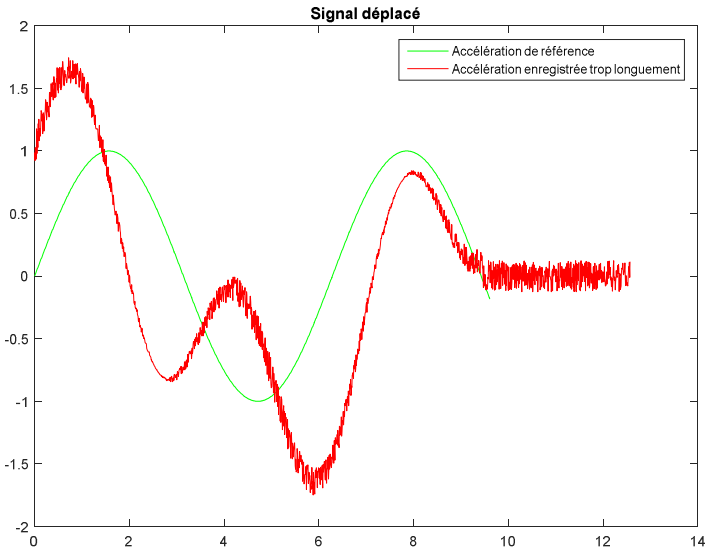
Reprenons le cas étudié à la partie précédente, pour lequel l'algorithme est performant.





Si la valeur de seuil était mal défini, ces différents cas peuvent être observés :

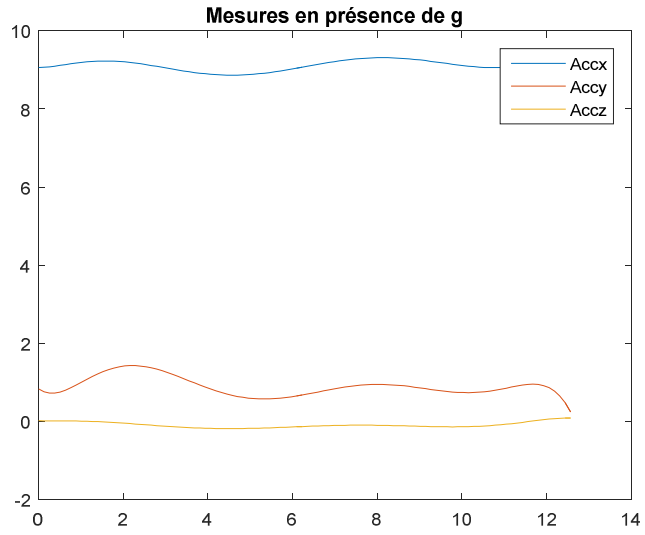


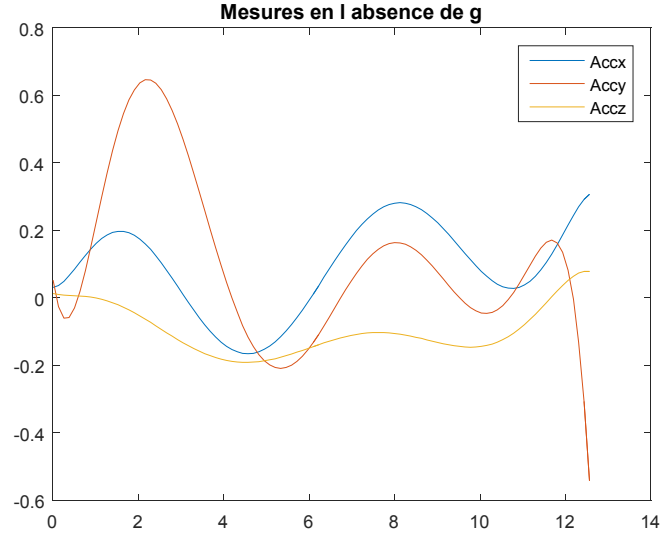


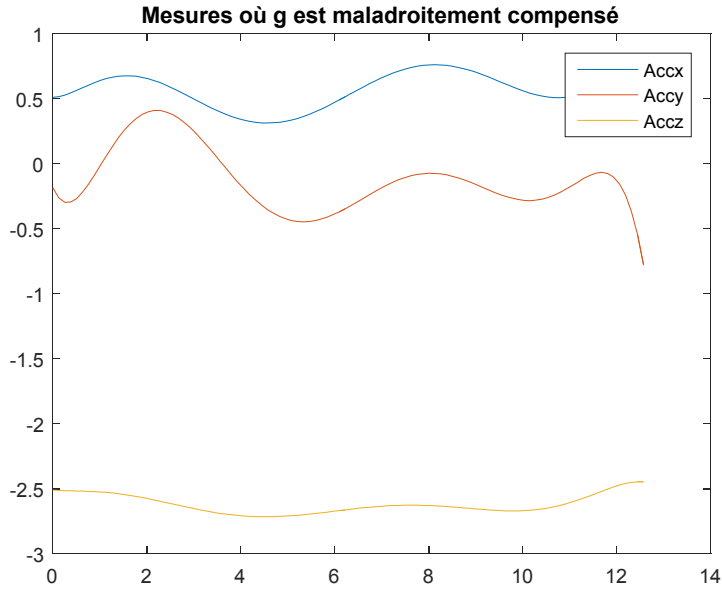
Dans le premier graphe, on remarque qu'en enregistrant le signal tardivement, on perd non seulement une partie des données utiles mais on corrompt aussi les suivantes en plaçant les signaux en opposition de phase. Dans le second graphe, l'arrêt de l'enregistrement se faisant tardivement, la fin de la mesure est incohérente avec la classe.

BRISSONNEAU Nicolas
CARRAND Adrien

Reprenons le sujet de la gravité, lorsque cette dernière est compensée correctement, on retrouve :







Dans le cas contraire, on peut se trouver confronté aux mesures suivantes :

Pour ce cas de figure, un retranchement maladroit de g=9,81m/s² se traduit d'une manière générale par un aplanissement des accélérations réelles du stylo. Effectivement, si la gravitée n'est pas entièrement compensée sur son axe alors qu'elle vient d'être soustraite, cela signifie qu'elle a été retranchée sur les autres accélérations, dans lesquelles se situaient les informations liées aux déplacements réels du stylo.

Pour résumer :

- La valeur de seuil peut corrompre la classe en rajoutant des données qui ne sont pas utiles ou en omettant celles qui le sont.
- La présence de la gravité a tendance à « noyer » les caractéristiques des courbes d'accélérations qui permettent de les identifier, assimilant fortement les acquisitions aux seules variations angulaires du stylo lié à la direction de la gravité.

Résolution des problématiques

Clarté du code

Lorsque les problématiques se multipliaient et que les origines des nouveaux problèmes nous étaient encore inconnues, nous nous sommes rendus compte que nous passions une quantité de temps colossale à nous retrouver dans notre propre travail. Nous avons alors raisonné de la manière suivante, nous nous sommes posés la question : si le projet nous était inconnu, serait-on capable de comprendre le fonctionnement global l'algorithme rapidement ? Si non, cela signifie qu'il est mal écrit. Si oui, cela signifie que nous sommes efficaces et sûrs de ce que nous entreprenons.

Pour rendre notre code Matlab plus accessible et plus compréhensible, nous avons repris sa rédaction. Le premier réflexe a été de fixer les paramètres, notamment ceux liés au calibrage du stylo, et de les placer dans des sections relatives à leurs fonctions. Cependant le problème de lisibilité était toujours important, nous avons alors continué la transformation en s'imposant une structure claire, simple, et en incorporant les principales étapes du code dans des fonctions. Cette étape fut un succès total : le code final peut tenir sur la seule fenêtre de Matlab et apporte beaucoup de clarté et de visibilité sur les problèmes rencontrés.

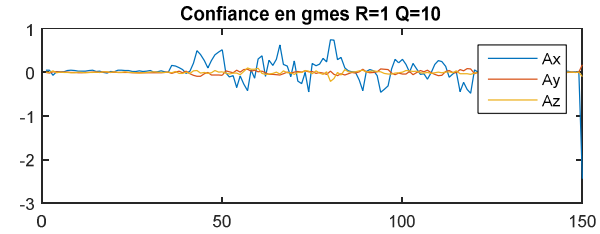
Debugge

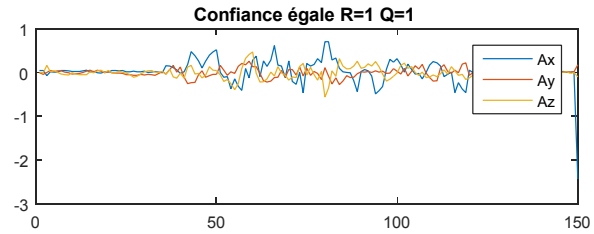
La réécriture du code nous a permis de rentrer dans une phase de debugge, l'objectif de cette phase était de lister l'ensemble des obstacles susceptibles de compromettre le bon fonctionnement de l'algorithme de reconnaissance de forme et de trouver des méthodes d'analyse efficaces pour répondre aux problèmes posés.

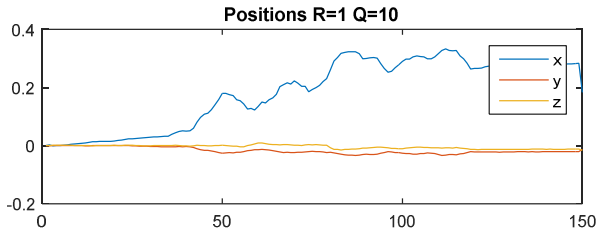
Le problème majeur auquel nous étions confrontés est que nous manipulions des accélérations, ce qui n'est pas très intuitif, encore moins quand elles sont bruitées et entachées de la présence de la pesanteur. Puisque nous mettions en doutes notre manière de traiter le signal au vu de nos premiers résultats, nous nous sommes repenchés sur le problème du traitement du signal pré-DTW.

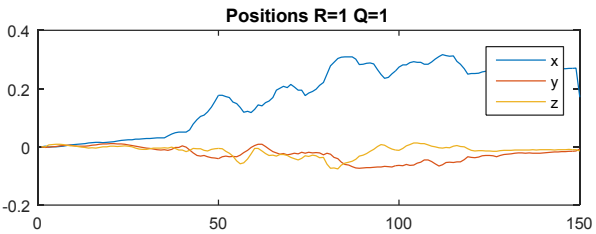
Pour savoir si nos signaux avaient encore du sens après les filtres appliqués, la solution la plus simple était d'intégrer deux fois, au moyen d'une double somme, afin de revenir aux déplacements. Ainsi, nous pouvions voir très simplement si nous avions corrompu le signal ou si nous l'avions épuré.

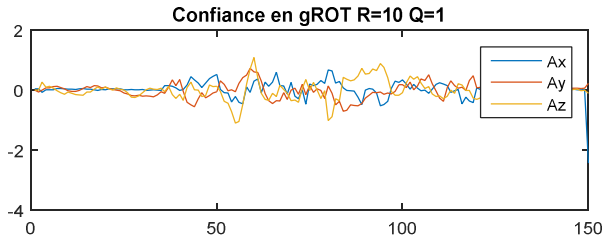
Reprenons l'exemple du signal montré dans la parties *prétraitement des données*. En lui retranchant g calculé là l'aide du filtre de Kalman expliqué précédemment, et en l'intégrant deux fois, nous obtenons les graphes suivant à partir du code DataExploited.m :

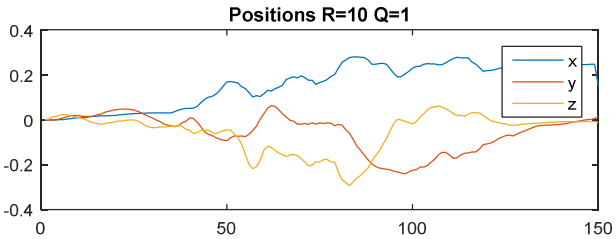






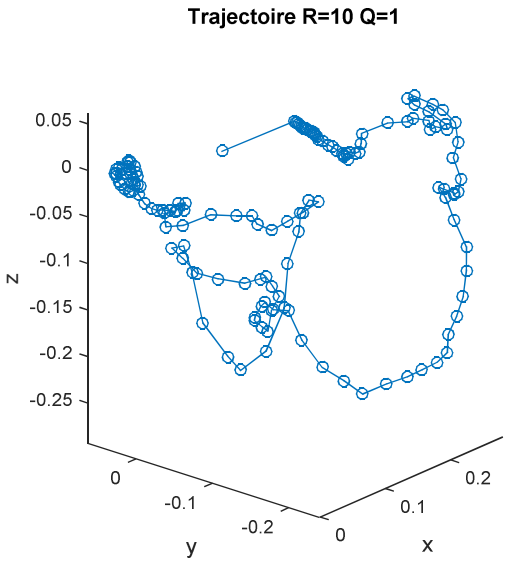


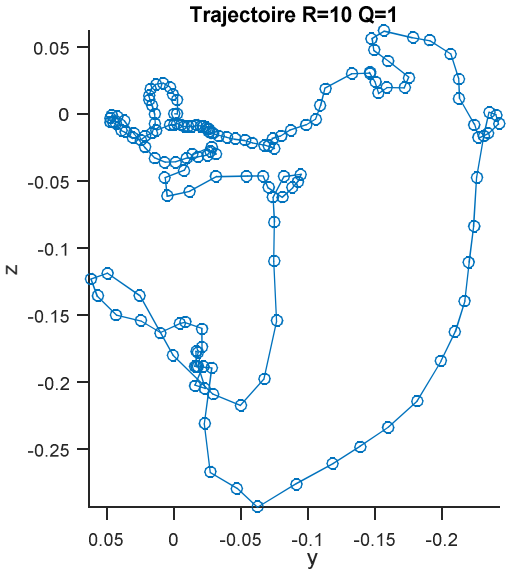




Sachant que le dessin réalisé avec le stylo pour obtenir ces signaux est un cercle, le premier graphe semble peu probable. Il écrase complètement Y et Z, sachant que la direction de la gravité était principalement sur X. Ces résultats donnent un stylo qui se déplace en X, et très peu en Y et Z, soit exactement l'inverse du résultat souhaité.

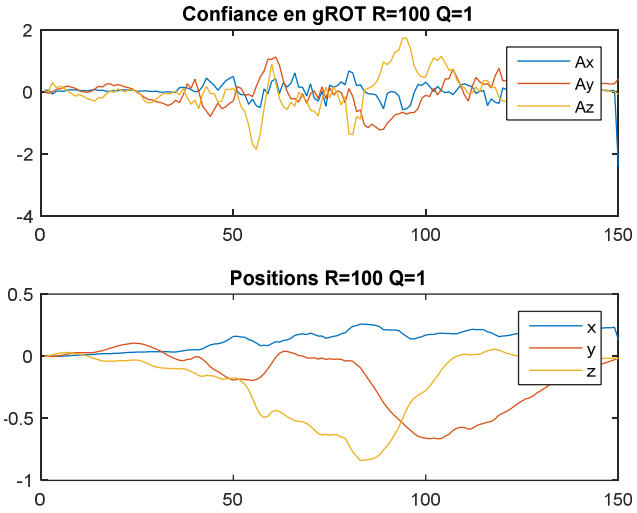
Les courbes suivantes sont plus intéressantes. On remarque que plus R est grand, donc plus on fait confiance à la correction de la direction de g à l'aide de la matrice de rotation, plus les résultats sont vraisemblables. Nous avons donc poussé dans cette direction en continuant à augmenter R. Pour se rendre mieux compte du résultat, nous avons étudier les graphes en 3D :

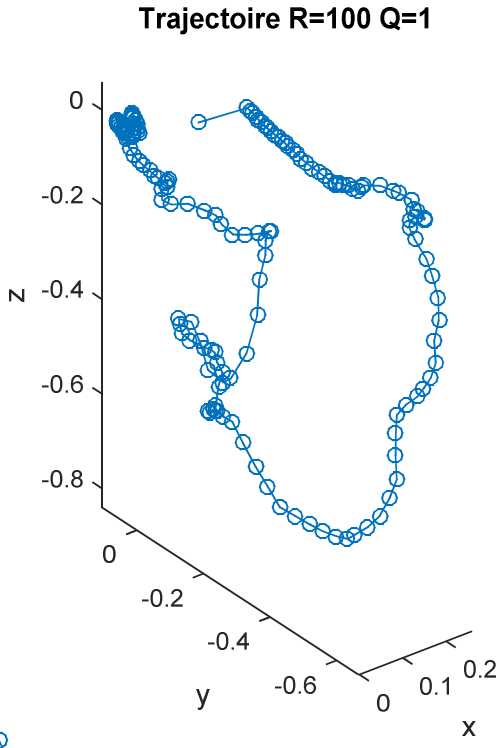


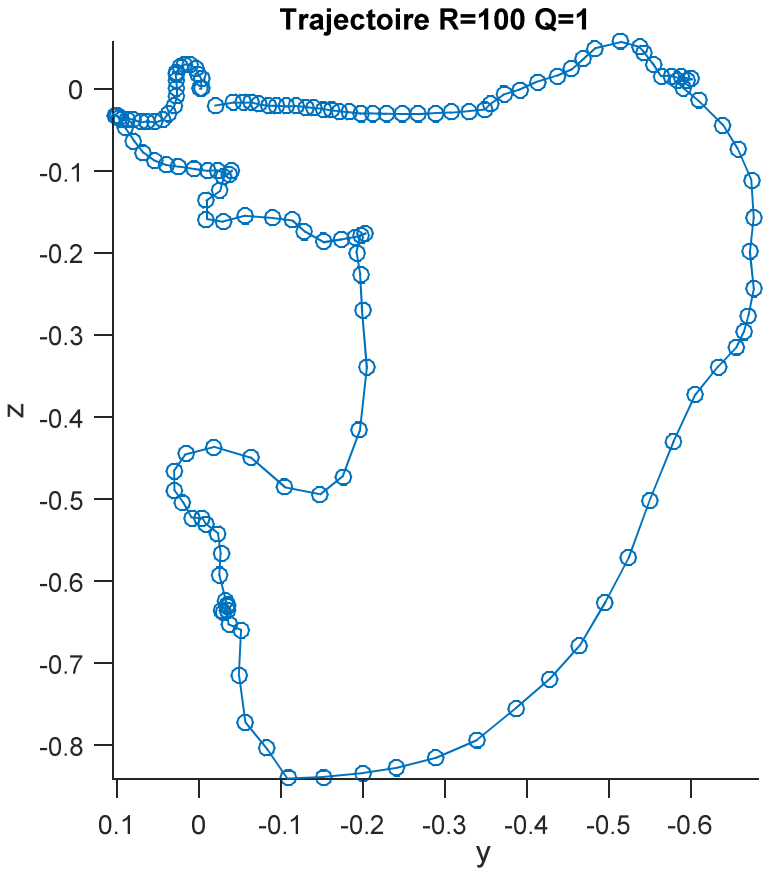


BRISSONNEAU Nicolas
CARRAND Adrien

On place le résultat du graphe en 3D dans le plan normal à x, qui était la direction du stylo. On obtient donc un semblant de cercle, on décide donc de poser R=diag(100,100,100) et Q=diag(1,1,1) :







Nous sommes satisfaits de ce résultat. En effet, continuer à augmenter la valeur de R n'apporte plus grand-chose. Nous sommes donc maintenant convaincus que quand les accélérations sont importantes, comme ici, il est bien plus intéressant de faire confiance aux gyroscopes pour retrouver la direction de la pesanteur et la retrancher.

En revanche, lorsque le stylo est à l'arrêt, le bruit des mesures du gyroscope affecte l'estimation de g. A ce moment-là, il est plus intéressant de récupérer la direction de la pesanteur à partir de la direction de l'accélération mesurée par l'accéléromètre.

C'est dans cet esprit que nous avons ajouté le paramètre « maj » dans la fonction filtre de Kalman : cette variable passe à 1 lorsque le stylo est considéré à l'arrêt, et elle permet ainsi de réinitialiser la valeur du vecteur g dans la direction de l'accélération mesurée. Nous obtenons ainsi de bien meilleurs résultats.

BRISSONNEAU Nicolas
CARRAND Adrien

Recherche d'une méthode de définition de classe

On a vu précédemment que les conditions de création de classe n'étaient pas parfaites, en effet le moindre défaut sur le prétraitement des données peut avoir un impact considérable sur l'acquisition des données utiles à la classe. Si nous sommes parvenus à grandement améliorer ces conditions, elles restent imparfaites et nécessite une approche différente de celle de la moyenne. Nous avons étudié la possibilité de créer le modèle de la classe nous-même puis de le modifier en y incorporant les « défauts » typiques apportés par l'homme en y mêlant des acquisitions. Cependant cela implique un travail considérable à chaque création de classe, travail dont l'utilité ne vaut pas le temps consacré. Au contraire même, puisqu'il consiste en l'adaptation de l'utilisateur aux besoins de l'algorithme, ce qui constitue une terrible interface homme-machine.

Nous avons abouti à la solution que nous considérons la plus pratique pour l'utilisateur, nous savons d'une part qu'il est rapide et simple d'effectuer des acquisitions et nous savons aussi que le DTW est un algorithme optimal en temps de calcul. Ainsi nous sommes arrivés à la conclusion que pour générer une classe cohérente il suffisait de produire une quantité importante d'acquisitions, puis de tester chacune entre elles pour repérer celles qui sont les plus reconnues les unes par toutes les autres. Cela donne un ensemble réduit de solutions que l'on peut alors manipuler plus aisément si l'on souhaite les moyenner ou les compléter.

Résultats Finaux

Nous sommes parvenus à reconnaître et à différencier des formes géométriques tels que des cercles et des carrés, mais aussi à écrire des chiffres. La reconnaissance est très bonne lorsque la personne qui a créé les classes manipule le stylo, mais elle se détériore lorsqu'un autre utilisateur tente sa chance. De plus, il faut noter que les conditions expérimentales sont particulières, on essaie d'écrire sur un plan perpendiculaire au sol, comme si l'on s'appuyait sur un tableau invisible. Le stylo est le plus stable possible durant l'écriture, et on essaie de faire correspondre le plus possible les temporalités classe/acquisition.

Ces conditions particulières se justifient par la démarche détaillée en début de rapport sur la faisabilité du projet. En effet, il était plus important pour le projet de prouver la faisabilité de ce dernier ainsi que la cohérence de l'ensemble de l'algorithme avec des résultats sous conditions expérimentales, plutôt que d'approfondir les méthodes de calcul pour chercher à obtenir l'ensemble des résultats simultanément au risque de n'en avoir aucun.

Du point de vue de l'apprentissage, bien que la méthode soit encore optimisable, nous sommes actuellement capables d'apprendre une forme simple au stylo en une minute. La qualité des acquisitions est telle qu'une simple acquisition peut servir de classe sans traitement, bien qu'elle montre un taux de de bonnes réponses légèrement plus faible.

Conclusion

Etendue de faisabilité/limites

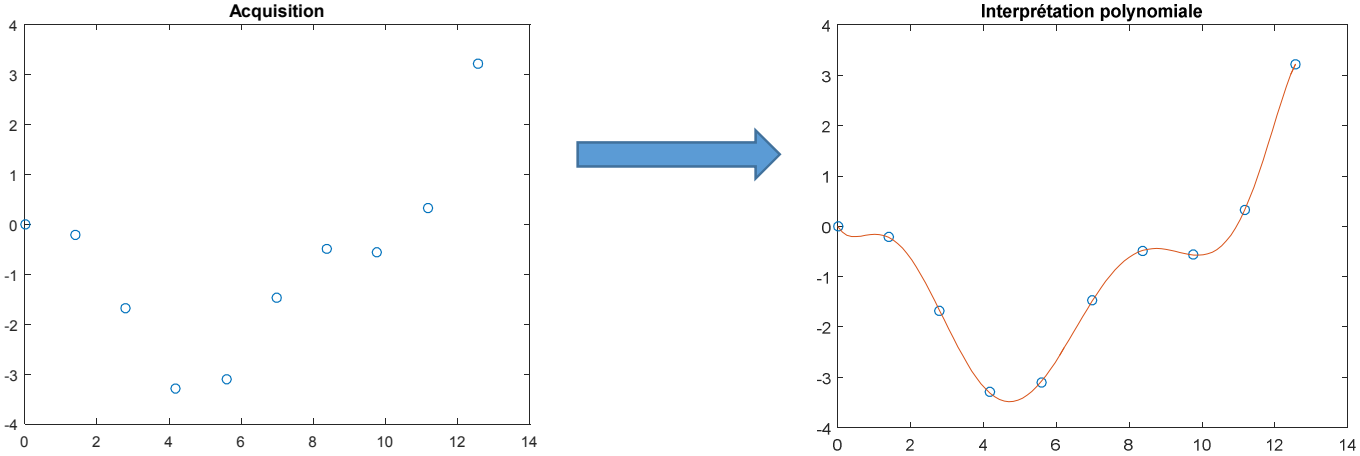
Si le stylo intelligent est capable de reconnaitre des formes simples telles que des chiffres et des formes géométriques, il reste contraint dans son utilisation par la performance technologique de ses composants mais peut aussi s'améliorer grâce à son utilisateur :

- On note des limites en sensibilité et en temps de réponse du système qui pénalisent la perception de mouvements subtils. L'écriture sur papier à vitesse réelle est donc prohibée pour le moment.
- Les performances du stylo varieront d'un utilisateur à l'autre, mais il est possible de conférer au stylo la capacité d'adaptation. Il pourrait donc adapter les interprétations de ses librairies en fonction de son propriétaire.
- Le stylo a tout ce qu'il lui faut pour apprendre rapidement de nouveaux patterns, son potentiel de faisabilité est donc extrêmement vaste !

Améliorations envisagées

Mode de lecture des données

Un sujet intéressant à développer serait l'interprétation des données, d'une part à travers le Machine Learning avec le DTW, mais aussi avec un prétraitement qui pourrait aider à cette reconnaissance. On peut penser à une interprétation polynômiale qui pourrait dans un même temps balayer la problématique de la base de temps fixe.



Cependant, les conditions dans lesquelles les mesures ont été effectuées montraient qu'un tel traitement des données n'était pas nécessaire à la démonstration de la faisabilité du projet, ainsi nous avons laissé ce développement de côté afin de favoriser l'obtention de résultats avant un tel travail d'optimisation.

Montage du stylo

Un autre point intéressant à développer est l'évolution du prototype : nous sommes conscients que ce dernier est encombrant et est contraint par son câblage. On peut alors proposer d'adapter le matériel aux besoins du stylo, en personnalisant le microcontrôleur de manière à ce qu'il tienne dans le gabarit d'un stylo, en ajoutant une connexion wifi et une batterie afin de lui conférer une liberté et une autonomie supérieure.

Bibliographie

Online Handwriting Recognition Using an Accelerometer-Based Pen Device , Jeen-Shing Wang, Yu-Liang Hsu, Cheng-Ling Chu, 2nd International Conference on Advances in Computer Science and Engineering (CSE 2013)

Estimation et Identification Partie II Estimation et Filtrage Optimal, Nazih Mechbal, Cours Arts et Métiers Paris 2015-2016

Annexes

MEMS-9DOF-VectorData.ino

```
//MPU-6050 and HMC5883L exploitation sketch
//By Brissonneau Nicolas and Carrand Adrien
//From Arts et Métiers School of Engineering, Paris, FRANCE
//April 13, 2016.
//Inspired by Arduino User JohnChi

#include <Wire.h> //I2C Arduino Library
#define address 0x1E //0011110b, I2C 7bit address of HMC5883L
const int MPU_addr = 0x68; // I2C address of the MPU-6050
int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ; //Acceleration, Temperature and
gyro
int16_t x,y,z; //Direction of g (gravitation)
int16_t tf; //length of a loop
int16_t tmax;


void setup() {
  //Initialize Serial and I2C communications
  Serial.begin(9600);
  Wire.begin();

  //MPU-6050 Initialization
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);

  //HMC5883L Initialization
  Wire.beginTransmission(address); //open communication with HMC5883
  Wire.write(0x02); //select mode register
  Wire.write(0x00); //continuous measurement mode
  Wire.endTransmission();

  tmax=41;
}

void loop() {
  tf=millis();

  //MPU-6050 Acquisition
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true); // request a total of 14 registers
  AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C
(ACCEL_XOUT_L)
  AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E
(ACCEL_YOUT_L)
  AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40
(ACCEL_ZOUT_L)
  Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42
(TEMP_OUT_L)
  GyX = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44
(GYRO_XOUT_L)
```

```
  GyY = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46
(GYRO_YOUT_L)
  GyZ = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48
(GYRO_ZOUT_L)
  Serial.print(AcX);Serial.print(" ");
  Serial.print(AcY);Serial.print(" ");
  Serial.print(AcZ);Serial.print(" ");
  //Serial.print(Tmp/340.00+36.53);Serial.print(" ");  //We don't use the
temperature
  Serial.print(GyX);Serial.print(" ");
  Serial.print(GyY);Serial.print(" ");
  Serial.print(GyZ);Serial.print(" ");

//  //HMC5883L Acquisition
//  Wire.beginTransmission(address);
//  Wire.write(0x03); //select register 3, X MSB register
//  Wire.endTransmission();
//  Wire.requestFrom(address, 6);
//  if(6<=Wire.available()){
//    x = Wire.read()<<8; //X msb
//    x |= Wire.read(); //X lsb
//    z = Wire.read()<<8; //Z msb
//    z |= Wire.read(); //Z lsb
//    y = Wire.read()<<8; //Y msb
//    y |= Wire.read(); //Y lsb
//  }
//  Serial.print(x);
//  Serial.print(" ");
//  Serial.print(y);
//  Serial.print(" ");
//  Serial.print(z);
//  Serial.print(" ");

  //Acquisition length
  tf=millis()-tf;

  //Make sure acquisition last at least for a certain amount of time
//  if (tf>tmax)
//  {
//    Serial.print("XXXXXXX");
//  }
//
  while (tf<tmax)
  {
    delay(1);
    tf=tf+1;
  }

  Serial.println(tf);
}
```


BRISSONNEAU Nicolas
CARRAND Adrien

KalmanG.m

```
function [ VtempS , gxyz , Pgxyz , VtempG , VtempROT , VgxyzROT] = KalmanG( Vtemp ,
gxyz , Pgxyz , te , H , C , Q , R)
% KalmanG : retranche l'acceleration de la pesanteur eux acquisitions de
% l'accelerometre en calculant la direction de l'acceleration de la
% pesanteur à l'aide d'un filtre de Kalman et du gyroscope.
%
% Initialiser dans la fonction Main :
% gxyz=(9.81*[0 0 1])';
% Cette valeur est également en sortie, dans un
% soucis d'utilisation de cette fonction dans une boucle
% Pgxyz=1000*eye(3);
% Cette matrice est en sortie pour les mêmes raisons que pour gxyz
%
% H=eye(3) Cf cours de M. Mechbal.
% C=eye(3)
%
% R = Matrice 3x3, Confiance dans la mesure (acceleration dy syst re-normée à 9.81)
% Q = Matrice 3x3, Confiance dans le système (gxyz modifié à l'aide des angles
mesurés)

KROT=1; % Coeficient qui parmet d'ajuster les eventuelles erreurs d'étalonnage des
gyroscopes.

VtempS=0*Vtemp;
VtempG=0*Vtemp;
VtempROT=0*Vtemp;
VgxyzROT=0*Vtemp;
gxyzROT=0*gxyz;
gxyzROT(1)=Vtemp(1,1);
gxyzROT(2)=Vtemp(1,2);
gxyzROT(3)=Vtemp(1,3);
gxyzROT=(9.81/norm(gxyzROT))*gxyzROT;

for i=1:te

    rx=KROT*0.041*Vtemp(i,4); % Angle autour de X pendant 41ms
    ry=KROT*0.041*Vtemp(i,5); % Angle autour de Y pendant 41ms
    rz=KROT*0.041*Vtemp(i,6); % Angle autour de Z pendant 41ms

%     % Mrxyz=Mrx*Mry*Mrz
%     Mrxyz =[cos(ry)*cos(rz),-cos(ry)*sin(rz),sin(ry);...
%     cos(rx)*sin(rz) + cos(rz)*sin(rx)*sin(ry), cos(rx)*cos(rz) -
sin(rx)*sin(ry)*sin(rz), -cos(ry)*sin(rx);...
%     sin(rx)*sin(rz) - cos(rx)*cos(rz)*sin(ry), cos(rz)*sin(rx) +
cos(rx)*sin(ry)*sin(rz),  cos(rx)*cos(ry)];

    % Mrxyz=Mrz*Mry*Mrx Matrice de rotation testée et approuvée
    Mrxyz =[ cos(ry)*cos(rz), cos(rz)*sin(rx)*sin(ry) - cos(rx)*sin(rz),
sin(rx)*sin(rz) + cos(rx)*cos(rz)*sin(ry);...
    cos(ry)*sin(rz), cos(rx)*cos(rz) + sin(rx)*sin(ry)*sin(rz),
cos(rx)*sin(ry)*sin(rz) - cos(rz)*sin(rx);...
    -sin(ry),cos(ry)*sin(rx),cos(rx)*cos(ry)];


%     Mrxyz =-[0 , -rz , ry;...
%         rz , 0 , -rx;...
%         -ry , rx , 0];


    %%%%%%%------- DEBUG -------%%%%%%%
```


BRISSONNEAU Nicolas
CARRAND Adrien

```
    % Cette partie débug permet de vérifier que la sortie du filtre de
    % Kalman, VtempS, tend bien vers VtempG lorsque Q est très grand devant
    % R, ou tend bien vers VtempROT losque R est très grand devant Q.

    % Methode retranchement G simple
    gx=Vtemp(i,1)/norm(Vtemp(i,1:3));
    gy=Vtemp(i,2)/norm(Vtemp(i,1:3));
    gz=Vtemp(i,3)/norm(Vtemp(i,1:3));
    VtempG(i,1)=Vtemp(i,1)-9.81*gx;
    VtempG(i,2)=Vtemp(i,2)-9.81*gy;
    VtempG(i,3)=Vtemp(i,3)-9.81*gz;
    % VtempG est la valeur des accélérations en retranchant la pesenteur de
    % manière grossière, en supposant qu'elle est de même direction que
    % l'accélération mesurée par l'accéléromètre.

    %Méthode rentranchement de G rotation
    gxyzROT=Mrxyz*gxyzROT;
    gxyzROT=(9.81/norm(gxyzROT))*gxyzROT;
    VtempROT(i,1)=Vtemp(i,1)-gxyzROT(1);
    VtempROT(i,2)=Vtemp(i,2)-gxyzROT(2);
    VtempROT(i,3)=Vtemp(i,3)-gxyzROT(3);
    VgxyzROT(i,1)=gxyzROT(1);
    VgxyzROT(i,2)=gxyzROT(2);
    VgxyzROT(i,3)=gxyzROT(3);
    % VtempROT est la valeur des accélérations en retranchant la pesenteur
    % initialisé par gxyz et actualisée à chaque i par la matrice de
    % rotation.
    % Vgxyz est la valeur de la pesenteur exprimée dans le repère local,
    % initialisé par gxyz et ectualisé à chaque i par la matrice de
    % rotation.



    %%%%%%%------- KALMAN -------%%%%%%%%



    % Récupération de la matrice de rotation
    % [rx,ry,rz]=0.053*[Vtemp(i,4),Vtemp(i,5),Vtemp(i,6)];

    % "Mesure" de g à travers le vecteur acceleration de l'ensemble
    norme=norm(Vtemp(i,1:3));
    gmes=((9.81/norme)*[Vtemp(i,1),Vtemp(i,2),Vtemp(i,3)])';

    % Prédiction
    xkk_=Mrxyz*gxyz;
    Pkk_=Mrxyz*Pgxyz*Mrxyz'+H*Q*H';

    % Correction
    K=Pkk_*C'*inv(C*Pkk_*C'+R);
    gxyz=xkk_+K*(gmes-C*xkk_);
    Pgxyz=(eye(3)-K*C)*Pkk_;

    % re-normage de gxyz à 9.81

    gxyz=(9.81/norm(gxyz))*gxyz;

    % Retrait de l'acceleration de la gravité des accélérations mesurées :
    VtempS(i,1)=Vtemp(i,1)-gxyz(1);
    VtempS(i,2)=Vtemp(i,2)-gxyz(2);
    VtempS(i,3)=Vtemp(i,3)-gxyz(3);

end
```


BRISSONNEAU Nicolas
CARRAND Adrien

DataExploited.m

```
% Parametres
chemin='C:\Users\Adrien\Desktop\Database';
d=dir(chemin); % Accès au répertoire
N=150; % Taille de la base de données
Te=4; % Pas de la moyenne glissante

% Opening File
%filename=fullfile(chemin, sprintf('Database_%d_%d.csv',N,1));
filename=fullfile(chemin, sprintf('Database_150_1.csv'));
Data=csvread(filename);
% Data=Data(20:end,:); % Les premières valeurs sont souvent faussées
N=length(Data);
t=(1:N);

Vtemp=Data;

% Plot initial values
figure(1)
%subplot(NT,1,1)
plot(t,Vtemp(:,1),t,Vtemp(:,2),t,Vtemp(:,3));
title('Accélérations brutes');
legend('Ax','Ay','Az');

figure(2)
plot(t,Vtemp(:,4),t,Vtemp(:,5),t,Vtemp(:,6));
title('Gyroscope brute');
legend('Gx','Gy','Gz');

% Low Filter
for j=0:N-Te
    Vtemp(N-j,1:3)=(sum(Vtemp((N-j)-(Te-1):N-j,1:3)))/Te;
end

figure(3)
plot(t,Vtemp(:,1),t,Vtemp(:,2),t,Vtemp(:,3));
title('Accélérations filtrées');
legend('Ax','Ay','Az');

for j=0:N-Te
    Vtemp(N-j,4:6)=(sum(Vtemp((N-j)-(Te-1):N-j,4:6)))/Te;
end

figure(4)
plot(t,Vtemp(:,4),t,Vtemp(:,5),t,Vtemp(:,6));
title('Gyro filtrées');
legend('Gx','Gy','Gz');

% R=100, Q=1
gxyz=(9.81*Vtemp(2,1:3))'/norm(Vtemp(2,1:3));
Pgxyz=1000*eye(3);
R=100*eye(3); %Confiance dans la mesure (acceleration dy syst re-normée à 9.81)
Q=1*eye(3); %Confiance dans le système (gxyz modifié à l'aide des angles mesurés)
[ Vtemp2 , gxyz , Pgxyz ,VtempG,VtempROT,VgxyzROT ] = KalmanG( Vtemp , gxyz , Pgxyz
, N , H , C , Q , R);

figure(8)
subplot(2,1,1)
plot(t,Vtemp2(:,1),t,Vtemp2(:,2),t,Vtemp2(:,3));
title('Confiance en gROT R=1000 Q=1');
legend('Ax','Ay','Az');

% DOUBLE INTEGRALE
XYZ=zeros(length(t),3);
```

```
VXYZ=zeros(length(t),3);
VXYZ(1,:)=Vtemp2(1,1:3)*0.041;
for i=2:length(t)
    VXYZ(i,:)=VXYZ(i-1,:)+Vtemp2(i,1:3)*0.041;
end
XYZ(1,:)=VXYZ(1,1:3)*0.041;
for i=2:length(t)
    XYZ(i,:)=XYZ(i-1,:)+Vtemp2(i,1:3)*0.041;
end

figure(8)
subplot(2,1,2)
plot(t,XYZ(:,1),t,XYZ(:,2),t,XYZ(:,3));
title('Positions R=100 Q=1');
legend('x','y','z');
figure(10)
plot3(XYZ(:,1),XYZ(:,2),XYZ(:,3),'-o');
legend('x','y','z'); xlabel('x'); ylabel('y'); zlabel('z'); axis equal;
title('Trajectoire R=100 Q=1');
display('3Dploted');
```

main.m

```
% clc
% clear all
close all
arduino=serial('COM3','BaudRate',9600);
disp('COM3 : opened')

% Calibration Manuelle Fin
load('matlab-nouvelle-calib.mat')
[B,SF] =
Calibration(CalibVxMax,CalibVxMin,CalibVyMax,CalibVyMin,CalibVzMax,CalibVzM
in);

% Open
disp('Acquisition : started')
fopen(arduino);

% Open WORD
word = actxserver('Word.Application');
word.visible=1; % Pour que Word s'ouvre en parallèle de MATLAB
document = word.Documents.Add;
selection = word.Selection;

% initialisation
te=10; % lecture
N=1; % lissage des données
n=3; % Nombre de points pris en compte pour jauger la variation de norme
(>1)

% Liste des paramètres
[ecart,Vectf,Vectf2,Vect,Vect2,Vtemp,Vtemp2,N,t,t1,tu,Af,Aftemp,zer,normete
mp,j,detect1,detect2,d1,g,n,H,C,R,Q,Pgxyz,gxyz,Vminmaxf,moy_norm,Vgxyzlist,
Vgxyz,Vgtemp,sav,t2,t3] = Parameters( te,N,n );

% Plots
[ h1,h2,h3 ] = drawing( Vectf,Vectf2,Vgxyzlist,t,sav );
load('classeO.mat');
load('classecarre.mat');
load('classe1.mat');
load('classe2.mat');

while j<2000
    % Acquisition raw data
    for i=1:te
        V=fscanf(arduino,'%d');
        if length(V)>5
            Vtemp(i,:)=[V(1) V(2) V(3) V(4) V(5) V(6)];
        elseif i==1
            Vtemp(i,:)=[0 0 0 0 0 0];
        elseif i>1
            Vtemp(i,:)=Vtemp(i-1,:);
        end

        % Acquisition ajustée
        if Vtemp(1,1)==0 && Vtemp(1,2)==0 && Vtemp(1,3)==0
            zer=zer+1
        end
```


BRISSONNEAU Nicolas
CARRAND Adrien

```
        Vtemp(i,1:6)=(Vtemp(i,1:6)-B(1:6))./SF(1:6);
    end
    Vtempraw=Vtemp;

    % Filtre passe bas (Valeurs moyennées)
    [Vtemp,Vtemp2] = FiltrePB(Vtemp2,Vtemp,N,te,ecart);

    % Retranchement de g
    [Af , gxyz , Pgxyz, VtempG , VtempROT, Vgxyz] = KalmanG(Vtemp , gxyz ,
Pgxyz , te , H , C , Q , R,Vgxyz(end,:)', maj);

    % Threshold
    [moy_norm,normetemp,maj] = threshold(Af,n,normetemp);

    % Normalisation
    N01 = normalisation(Af,Vminmaxf,normetemp(1),detect1);

    % Actualisation du plot
    [Vectf,Vectf2,sav,t3] =
ActuPlot(ecart,Vectf,N01,Vectf2,Vtemp,h1,h2,h3,t,te,sav,t1,t2,t3,Vtempraw,V
gxyzlist,Vgxyz);

    % Détection de mouvement
    if (abs(normetemp(1))>detect1 & abs(moy_norm)>detect2) % Sélectionner
les points
        tu=tu+1;
        if t1==1
            t2=1;
        end
        t1=1;
    elseif (abs(normetemp(1))<=detect1 | abs(moy_norm)<=detect2) & t1==1
        % DTW

        [Dist,N1,N2]=dtw2(Vectf(end-((tu+1)*te-1):end-te,:),Un,Deux)

         if (N1<N2) & (N1 < 1)
             selection.TypeText('1');
             selection.TypeParagraph;
         elseif (N1>N2) & (N2 < 1)
             selection.TypeText('2');
             selection.TypeParagraph;
         end
        tu=0;
        t1=0;
        t2=0;
    end
    j=j+1;
end

% Close
fclose(arduino);
disp('Closing arduino')
delete(instrfindall)
clear arduino
```


BRISSONNEAU Nicolas
CARRAND Adrien