Rose Many Bella

Quénée Emile

Projet SI 380

Reconnaissance de mouvement

Encadrant :

Pascal Bianchi & Slim Essid

Table of Contents

[Introduction 2](#_Toc391387633)

[Contexte 2](#_Toc391387634)

[Problématique 2](#_Toc391387635)

[Définition du problème 2](#_Toc391387636)

[Framework 3](#_Toc391387637)

[Capteurs 3](#_Toc391387638)

[Communication 3](#_Toc391387639)

[Application android 3](#_Toc391387640)

[Architecture logicielle 3](#_Toc391387641)

[Logique de fonctionnement de l’architecture logicielle 4](#_Toc391387642)

[Détection de mouvement 5](#_Toc391387643)

[Définition du mouvement 5](#_Toc391387644)

[Qualification du mouvement 6](#_Toc391387645)

[Résolution de l’équivalence des déplacements par homothétie 6](#_Toc391387646)

[Résolution de l’invariance par rotation autour de l’axe y 7](#_Toc391387647)

[Détection du mouvement 8](#_Toc391387648)

[En pratique 9](#_Toc391387649)

[Reconnaissance de mouvement 11](#_Toc391387650)

# Introduction

## Contexte

L'objectif de ce projet est la reconnaissance de mouvement. Le sujet étant vaste, nous nous concentré sur un type de mouvement en particulier : les mouvements de type chorégraphique. Prenons un danseur. L'idée serait à partir des mouvements effectués d’interagir avec la musique qui suivrais ainsi la danse. La première question technique qui se pose est comment faire l’acquisition du mouvement ? Le partie a été de faire une reconnaissance embarqué sur le danseur à partir d’accéléromètres, et plus précisément à partir du capteur d'accélération embarqué sur la plupart des Smartphones android. Cela pour une raison simple : cela facilite le développement car on peut profiter de l'API pour récupérer les données et les envoyer vers un ordinateur, et permet de faire une application plus facilement réutilisable.

## Problématique

Le contexte général étant posé il nous reste maintenant à savoir comment effectuer la reconnaissance de mouvement, et quel type de reconnaissance en particulier nous voulons développer pour notre application. Plus concrètement nous cherchons à définir ce qu'est le mouvement dans notre problème, ainsi que les invariances, c'est à dire ce à quoi nous ne voulons pas être sensible.

## Définition du problème

Ainsi pour notre projet nous somme fixé les contraintes suivante :

* Le danseur tiendra le Smartphone dans la main, le poignet étant dans le même sens que l'axe y du repère du Smartphone (axe de la longueur)
* Nous voulons être indépendant de l'angle que fait le poignet autour de l'axe y : des mouvements ne différent que par la valeur de cet angle seront considérés comme identiques

Nous appellerons ainsi **mouvement** de manière idéal la courbe décrite dans l'espace par le déplacement de la main du danseur, indépendamment de l'angle que fait le poignet autour de l'avant bras. On précise cependant que cet angle peut varier pendant le mouvement, l'invariance étant vis à vis de la positon initiale de la main.

Finalement nous voulons que cette **reconnaissance** se fasse en **continue**, sans que l'utilisateur aie à spécifier quand est ce que démarre un mouvement est quand est ce qu'il s'arrête.

De ces considérations nous avons articulé notre projet en deux composantes majeurs, qui sont la **détection et traitement** du mouvement et la **reconnaissance** en tant que tel. Mais avant de rentrer dans le vif du sujet, tout ce travail repose sur un Framework qu’il conviens de décrire car ce dernier constitue la base de travail sur le projet .

# Framework

## Capteurs

Nous disposions pour le projet de deux capteurs d’accélérations  :

* Une Wiimote, possédant une étendue d’acquisition de , de fréquence d’échantillonnage
* Un Smartphone android (Samsung Galaxy S3), possédant une étendue d’acquisition de , de fréquence d’échantillonnage

La reconnaissance a été développé pour le Smartphone, la Wiimote ayant été utilisé pour effectuer des tests.

## Communication

La communication entre les capteurs et l’ordinateur qui s’occupait des traitements a été faite en Bluetooth

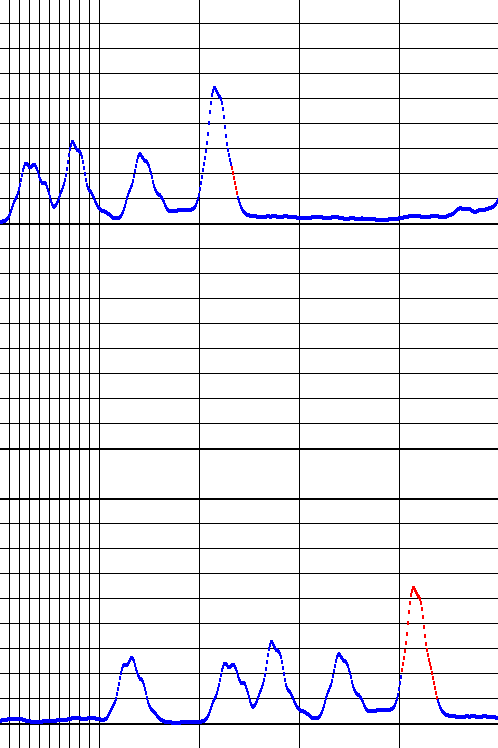
## Application android

Une application a été développé pour manipuler aisément l’acquisition des données d’accélérations et la communication avec l’ordinateur. Celle-ci comprend les fonctionnalités suivantes :

* Choix du périphérique Bluetooth pour l’interfaçage, et connexion
* Un bouton principal pour lancer ou non l’acquisition de l’accélération
* Un bouton pour spécifier au logiciel si on souhaite ou non faire de la reconnaissance en continue
* Un bouton pour dire au logiciel si les mouvements effectués ont pour vocation d’être appris par l’application pour une reconnaissance future
* Une entrée clavier pour spécifier le nom du mouvement que l’on souhaite faire apprendre

## Architecture logicielle

Une architecture logicielle complète a été développé en python pour communiquer avec l’utilisateur de manière ergonomique et fonctionnelle. Celle-ci comprend les fonctionnalités suivantes :

* Server Bluetooth pour accepter les connexions des capteurs (possibilité de plusieurs connexions simultanées)
* Gestion de la communication avec l’établissement d’un protocole de communication permettant de faire dialoguer le programme avec l’application de manière à tout décider depuis le Smartphone (mode acquisition continue et mode apprentissage)
* Un mode apprentissage permettant à la volée de faire apprendre à l’application de nouveaux mouvements
* Une gestion de la base de données de mouvement
* Un module d’affichage des données permettant un monitoring des différentes variables utilisés dans les algorithmes (représentation ci-contre). Comme ce module va être utilisé à de nombreuses reprises pour illustrer des résultats il convient de spécifier la valeur des axes qui seront inchangé par la suite:
  + Les axes horizontaux ont pour espacement la valeur 1, le trait plus gras central correspondant à la valeur 0. Cela donne une idée de l’amplitude du signal.
  + Les axes verticaux ayant pour espacement une valeur de 100 (soit 100 échantillons d’accélération par exemple), avec une graduation plus courte sur la gauche avec un trait tous les 10 échantillons. Cela donne une idée de la taille du signal.

Ce module permet de tracer plusieurs signaux en simultanés que l’on peut spécifier à la volé.

* Un module midi permettant de faire un lien entre la base de donnée de mouvement et des notes midis pour ensuite interfacer le logiciel avec un programme de production musicale, pour en finalité interagir avec de la musique grâce aux mouvement reconnus.

Et surtout :

* Un module de détection de mouvement avec un prétraitement des données
* Un module de reconnaissance du mouvement

Brossons rapidement la logique de fonctionnement de cette architecture

## Logique de fonctionnement de l’architecture logicielle

Les accélérations sont extraites en continue du Smartphone et envoyé vers le Smartphone. A partir d’ici nous avons trois cas de figure :

* L’application n’est ni en mode apprentissage ni en mode acquisition continue et le mouvement est simplement détecté par la fin de l’envoie de données. Cette fonctionnalité a été la première développé avant de s’intéresser exclusivement à la reconnaissance en continue.
* L’application est en mode détection continue sans mode apprentissage. Ici l’application fonctionne en deux temps : lorsqu’un mouvement n’est pas détecté on chercher à évaluer la valeur de l’angle de rotation de poignet ainsi que la valeur d’accélération de la pesanteur dans le référentiel du Smartphone, que l’on vient soustraire au mouvement lorsque celui-ci est détecté. S’en suit une phase de reconnaissance ou le mouvement est comparé une base de donnée pré établie par l’utilisateur ou l’on va chercher à savoir quel est le mouvement qui lui ressemble le plus
* L’application est en mode apprentissage. Que l’on soit en mode détection continue ou non, lorsque qu’un mouvement est détecté celui-ci va être ajouté à la base de donnée associé à l’étiquette spécifié par l’utilisateur. Une même étiquette peut être utilisé pour plusieurs mouvements différents pour augmenter les chances d’une bonne reconnaissance.

Une fois un mouvement reconnue celui-ci est associé à un message midi selon une base de donnée, qui va être généré pour être envoyé vers un logiciel de production musicale pour interagir avec de la musique.

Maintenant que le contexte de travail a été abordé nous allons pouvoir nous intéresser à ce qui fait le cœur de ce projet, à savoir la détection et la reconnaissance de mouvement.

# Détection de mouvement

Avant de savoir comment repérer un mouvement, nous allons voir quel est la définition que l’on donne au mouvement, puis comment nous qualifions ce mouvement

## Définition du mouvement

L’accéléromètre du Smartphone nous donne de manière brute la valeur de l’accélération mesurée dans son référentiel.

L’accélération mesuré est de la forme suivante :

Avec :

* La projection de l’accélération de la pesanteur dans le référentiel du Smartphone
* L’accélération du mouvement effectué, qui est la composante cherché.

On considère en effet que l’utilisateur est un référentiel galiléen.

On précise dès maintenant que l’on considèrera deux référentiels :

* Celui du Smartphone ou les axes sont données ici en illustration.
* Le référentiel de l’utilisateur , où l’axe z est vertical, selon la direction de la pesanteur.

Ainsi pour reconnaître notre mouvement idéal il conviendrais de connaître avec précision la position du Smartphone dans le référentiel de l’utilisateur.

Pour cela la méthode analytique est de faire une double intégration du vecteur .

Cela entraîne cependant quelques complications :

* Comme la valeur de l’accélération n’est pas constante il n’est pas possible d’obtenir un formule exacte de la position en fonction de la valeur de l’accélération par des méthodes simples.
* Pour accéder à la valeur de il est nécessaire de retrancher à chaque instant, et il n’est pas évident de connaître la valeur de cette projection car nous ne contraignons pas le Smartphone à suivre des déplacement simple comme des homothéties ou des rotations uniquement suivant un axe. Les mouvements pouvant être à priori quelconques.

Ainsi vis-à-vis de ces complications nous considérerons un mouvement comme étant simplement une suite de valeurs d’accélérations.

Cela n’est cependant pas restrictif car la différence d’information entre la position et l’accélération sont deux constantes d’intégrations, qui sont la position et la vitesse à l’état initial. Hors dans notre cas ce sont deux informations qui nous intéresse pas :

* Premièrement nous souhaitons uniquement connaitre la courbe que décrit le mouvement, sans se soucier de sa position à l’origine que l’on peut fixer arbitrairement à valeur que l’on veut.
* Ensuite comme nous allons le voir, nous allons considérer qu’il y a un mouvement lorsque la valeur de la norme de n’est plus négligeable devant celle de. Ainsi on peut considérer que la vitesse à l’origine du mouvement est nulle. C’est ce qui nous permet au passage de distinguer intuitivement les mouvements entre eux avant de penser à un algorithme.

Nous pouvons ainsi sans perte d’information considérer qu’un mouvement est la suite d’accélération décrite par le mouvement de la main de l’utilisateur. L’enjeu est à partie de de retrouver cette accélération.

## Qualification du mouvement

Cette considération faite il reste à savoir quels sont les différents types d’accélérations que peut subir le Smartphone. Elle sont principalement de 2 natures :

* Homothétie selon un axe
* Rotation autour d’un axe

Pour qualifier le mouvement nous considérons les invariances suivantes :

* Le déplacement par homothétie est équivalent en terme d’accélération pour tous les axes
* Insensibilité à la rotation autour de l’axe y

### Résolution de l’équivalence des déplacements par homothétie

Déplacement selon x, selon z

Cela n’a pas été une contrainte que l’on avait à l’esprit avant de voir les données. En effet lorsque l’on déplace le Smartphone selon la direction du vecteur on obtient une dynamique de variation de l’accélération (qui se projette sur cette direction, on est ici dans le référentiel de l’utilisateur) beaucoup plus importante que les autres.

On donne ici en illustration la norme du vecteur d’accélération , pour le Smartphone posé à plat sur une table. L’amplitude d’accélération n’est pas comparable que l’on se déplace selon l’axe x ou z dans le référentiel de l’utilisateur.

Pour obtenir à notre invariance il est nécessaire d’accéder uniquement à l’accélération

Pour se faire nous considérons qu’au repos ou pour des mouvements suffisamment lents, nous avons :

Une méthode pour recouper cette accélération à l’accélération mesurée va être de prendre comme vecteur une moyenne de l’accélération sur les derniers échantillons que l’on recoupe ensuite à .

Au repos l’accélération et ainsi nulle, et c’est l’écart à cette moyenne qui nous indique le mouvement. On remarque que cela revient à calculer la dynamique du mouvement car ce que nous effectuons en réalité est une différentiation du vecteur d’accélération.

**Formellement :**

étant l’opérateur de moyenne des derniers échantillons de

Une fois l’accélération du déplacement obtenue nous voulons que ce vecteur soit invariant par rotation autour de l’axe y.

### Résolution de l’invariance par rotation autour de l’axe y

Il suffit pour cela de corriger l’accélération par la matrice de rotation relative à cet axe :

Avec :

Ainsi :

Avec :

* Accélération du geste

**Soit finalement :**

## Détection du mouvement

Au repos le vecteur est de norme presque nulle, et c’est lorsque la valeur de sa norme dépasse un certain seuil que l’on peut dire qu’il y a un mouvement.

Détection du mouvement avec une seuil à 2

Cependant cette formule pour ne devient plus valide en temps réel :

* Le mouvement va avoir pour effet de perturber la valeur de
* L’égalité n’est plus valable en mouvement

Pour contrer ce problème nous allons au moment où le mouvement est détecté fixer ces paramètres, avec l’astuce de les retarder un peu dans le temps pour éviter d’être perturbé par le fait que la détection de mouvement se fait toujours avec une certaine latence, qui est celle d’atteindre le seuil.

Nous allons ainsi, au moment où le mouvement est détecté récupérer fixer la valeur de et , que l’on récupère de manière retardé, et se sont ces valeurs que l’on va utiliser pour calculer à partir des accélérations correspondant au mouvement :

Avec :

et fixé avec un retard de échantillons .

C’est cette accélération que l’on va utiliser pour effectuer la reconnaissance.

Les accélérations correspondants au mouvement sont :

* Celle qui s’effectue au dessus du seuil
* Quelques échantillons d’accélération pré seuil qui participent en réalité au mouvement

Et nous avons un avons un mouvement lorsque :

Nous avons ainsi grâce à cette partie théorique une formule donnant la valeur de en fonction de deux paramètres qui sont :

* Nombre d’échantillons servant à moyenner
* Le retard d’échantillons pris pour fixer et

Et un paramètre nous permettant de dire si oui ou non un mouvement est entrain d’être effectué.

## En pratique

Sans et avec un lissage sur une fenêtre de 15 échantillons de l’accélération et effet sur la norme

En pratique nous voulons un comportement de la détection qui soit cohérente vis-à-vis de l’utilisation et des conditions d’utilisations. Nous voulons notamment être suffisamment robuste aux faibles mouvements pour éviter de déclencher des reconnaissances de manière intempestive. L’idée est de pouvoir se déplacer ou de placer son bras sans que cela soit reconnue comme étant un mouvement. Nous voulons cependant garder un comportement suffisamment réactif pour donner l’impression d’une reconnaissance en temps réel.

La calibration des valeurs de l’algorithme proviennent ainsi de la fréquence d’échantillonnage du Smartphone qui est de 400 HZ ainsi que d’un ajustement par expérimentation pour obtenir un comportement cohérent vis-à-vis de son application.

Premièrement comme les données fournies par l’accéléromètre sont légèrement bruités nous allons effectuer une première étape de filtrage. Une taille de fenêtre de 15 échantillons est suffisamment petite pour obtenir un lissage satisfaisant pour supprimer le bruit sans prendre trop de latence. (On observe déjà ici un retard d’une dizaine d’échantillons pour la remise à 0 du signal). Ensuite une taille de 15 a également été choisie pour moyenner , et une valeur de 30 pour le retard . Le seuil quand à lui a une valeur de 2.

Robustesse de la reconnaissance avec un seuil 2 pour des perturbations suffisamment courtes.

A l’utilisation cependant deux considérations ont dues être rajouté pour obtenir le comportement souhaité :

* Une des plus grande cause de parasite sont les mouvements de rotation du poignet, du fait qu’ils font varier rapidement la valeur de l’accélération sur les différents axes. Pour se prémunir de cela une contrainte a été rajouté sur le mouvement : qu’il dure un certain nombre d’échantillon. En dessous d’une certaine longueur au dessous du seuil la perturbation n’est pas qualifié comme étant un mouvement
* Certains mouvement proche, pourtant mis en valeur par l’utilisateur en effectuant une pose non négligeable entre les deux mouvements pour les discerner intuitivement, ne permettaient pas de faire retomber la norme sous le seuil du fait de la latence introduite par les différentes étapes de lissage. Un remède à cela a été d’introduire une détection de pic min et max lors des phases de mouvement. Si au cour d’un mouvement une différence de pic suffisament grande est repéré on coupe le mouvement en deux au niveau de ce pic de faible intensité. La seuil de cette différence a été prise expérimentalement à 2.

Une différence suffisament importante entre un pic max et un pic max entraine la séparation en deux mouvements distincts

Le mouvement étant maintenant extrait, il reste à le reconnaitre.

# Reconnaissance de mouvement

* Introduction  
  - Problématique/ Définition du problème  
  - Méthodologie/ Architecture du système  
  - Travail effectué  
    - Invariance accélération  
    - Détection/extraction d'un  mouvement  
    - Reconnaissance de mouvement  
      - Prétraitement du vecteur d'accélerations  
      - Extraction des caractéristique/Features  
      - Classification  
        - Calcul de similarité: DTW  
        - Classifieur: KNN  
        - Optimisation classification: Kmeans(clustering) + KNN dans le centre de cluster le plus proche.  
    
  - Résultats/Démonstration  
  - Conclusion