Manuel de l'Utilisateur

Traitement de séquences et manette Wii

Nicolas CHEIFETZ
Supervisé par Thierry Artières

juin-juillet 2008

Laboratoire d'Informatique de Paris 6

Table des matières

1 Procédure d'installation **2**
1.1 Contraintes techniques . . . 2
1.2 Démarrer . . . 2

2 Paramètres de fonctionnement **3**
2.1 Exécuter les programmes "à la main" . . . 3
2.2 Remarques . . . 4

3 Fonctionnalités et interactions utilisateur **5**
3.1 Comment manipuler la manette ? . . . 5
3.2 `Create_bases.java` . . . 5
3.3 `Main.java` . . . 5
3.3.1 Exemple d'ajout de Features (cas 18) . . . 9
3.4 Construction des bases de séquences par Cross Training . . . 9
3.5 `TestDTW.java` et `TestCRF.java` . . . 10

Références **11**

Chapitre 1

Procédure d'installation

Le projet a été écrit en Java et destiné au système d'exploitation Windows XP. Il a été développé sous l'IDE Eclipse. Cependant, il n'est pas nécessaire d'installer Eclipse pour travailler sur ce projet. Si vous choisissez de l'ouvrir sous Eclipse, nous avions créer une procédure pour le PIAD n^{o}25[1].

1.1 Contraintes techniques

Les seules contraintes techniques sont :

- travailler sous le système d'exploitation Windows XP
- avoir installé une version récente du "Java Runtime Environnement" [2]
- disposer d'une connexion bluetooth : version 2.0 + EDR.

1.2 Démarrer

Pour travailler sur le projet, il vous suffit de vous placer dans le répertoire `Stage_final\start` et de double-cliquer sur les scripts batch qui s'y trouvent.

Chacun de ces scripts exécute une tâche bien précise :

`Main.bat` :	propose 20 fonctionnalités utiles au développement
`Create_bases_0.bat` :	crée des bases de données à partir d'une wiimote, construction des bases par Cross Training, et création des bases "croisées" sous un format lisible par CRFBranch
`Create_bases_1.bat` :	construction des bases par Cross Training, et création des bases "croisées" sous un format lisible par CRFBranch
`Create_bases_2.bat` :	création des bases "croisées" sous un format lisible par CRFBranch
`TestDTW.bat` :	test avec le modèle DTW au travers d'une Wiimote (appel au programme `TestDTW.java`)
`TestCRF.bat` :	test avec le modèle CRFBranch au travers d'une Wiimote
`TrainTestCRFBranch.bat` :	apprendre/tester un ou plusieurs modèle(s) CrfBranch

Pour plus d'informations, veuillez lire le Chapitre 3 et le Manuel du Programmeur[3].

[1] `Procédure pour travailler sur le piad sous Eclipse` : *ici*
[2] `par exemple` : *JRE 1.6*

Chapitre 2

Paramètres de fonctionnement

Lors de l'exécution des programmes, les fichiers compilés sont dans la librairie `LibStage.jar`. Le répertoire courant (dans lequel sont exécutés les programmes) se trouve dans le dossier : `Stage_final\Project\java\bin`.

Il existe cinq programmes java exécutables (qui contiennent une méthode `main()`) :

`Main.java` :	propose 20 fonctionnalités utiles au développement
`Create_bases.java` :	crée des bases de données à partir d'une wiimote, construction des bases par Cross Training, création des fichiers lisibles par CRFBranch
`TestDTW.java` :	test avec le modèle DTW au travers d'une Wiimote
`TestCRF.java` :	test avec le modèle CRFBranch au travers d'une Wiimote
`TrainTestCRFBranch.java` :	apprendre/tester un ou plusieurs modèle(s) CrfBranch

2.1 Exécuter les programmes "à la main"

Placé dans le répertoire courant, on exécute les fichiers java à l'aide de la commande :

```
java -classpath LibStage.jar <nom_du_fichier>
```

Toutefois, les programmes `TrainTestCRFBranch.java`, et `Create_bases.java` dérogent à la règle :

- `java -classpath LibStage.jar TrainTestCRFBranch <fichier_parametres_crfbranch>` , ici le fichier qui contient l'ensemble des paramètres (commentés) nécessaire au programme, est `crfbranch_parametres.txt`
- `java -classpath LibStage.jar Create_bases <fichier_noms_gestes> <cas> <nom_rep_dest>`, `<nom_rep_dest>` est le dossier dans `Stage_final\Project\Datas\CRFs` qui contiendra les bases "croisées" écrites au format CrfBranch (ici, c'est `CRF_WinXP`) ;
  `<fichier_noms_gestes>` est le nom du fichier qui contient (ici, c'est `noms_gestes.txt`) ;
  `<cas>` peut être instanciée par trois valeurs :
  - `0` : création de bases de données à partir d'une wiimote, construction des bases par Cross Training, et création des bases "croisées" sous un format lisible par CRFBranch
  - `1` : construction des bases par Cross Training, et création des bases "croisées" sous un format lisible par CRFBranch
  - `2` : création des bases "croisées" sous un format lisible par CRFBranch

Il est bien entendu que vous pouvez modifier les arguments des programmes à votre guise. Pour cela, il vous suffit de modifier les commandes en éditant les scripts batch (`Stage_final\start`) dans un éditeur de texte. Les fichiers textes `crfbranch_parametres.txt` et `noms_gestes.txt` sont dans le répertoire courant.

2.2 Remarques

La variable `buffer_length` est fondamental pour le bon déroulement de notre projet. Elle va de pair avec le concept de boîtes[1]. Cette variable est en général déterminée par la première boîte créée et sa valeur ne sera jamais modifiée au cours d'une tâche. Usuellement, on instancie la variable `buffer_length` par un entier entre 10 et 20.

Vous pouvez ajouter des commentaires dans les fichiers `<fichier_parametres_crfbranch>` et `<fichier_noms_gestes>`. Le symbole est %.

[1] Voir le Manuel du Programmeur[3] et le Dossier d'analyse et conception du PIAD[4]

Chapitre 3

Fonctionnalités et interactions utilisateur

3.1 Comment manipuler la manette ?

On rappelle que la Wiimote est connectée par bluetooth.
Une hypothèse forte pour faciliter la reconnaissance est la position de la manette entre les mains de l'utilisateur. On suppose que l'utilisateur la tient de manière fixe : la Wiimote au creux de la main et l'index positionné sur le bouton **B**.

L'allumage de la manette se fait par la pression des touches **1** et **2**. Et elle s'éteint en pressant le bouton **POWER**.

Le bouton **B** est réservé à l'enregistrement des séquences. Une première pression démarre l'enregistrement, la suivante stoppe l'enregistrement.

3.2 `Create_bases.java`

Ce programme affiche une interface agréable à l'utilisateur lorsque celui-ci crée ses bases de données. Tout au long de son échantillonage, l'utilisateur dispose du graphe simulant les accélérations au cours du temps, de la fenêtre décrivant les fonctionnalités des boutons et la fenêtre 3D. On remarque que durant l'échantillonage deux autres fenêtres aident l'utilisateur à savoir quel est le geste qu'il doit effectuer et combien en a-t-il enregistré. Ainsi, il n'est pas possible de se tromper de classes pour un geste !

La base enregistrée se trouve dans le répertoire `Stage_final/Project/Datas/Gestes` et prend le nom que l'utilisateur lui aura donné.

Enfin, il existe trois manière différentes d'exécuter ce programme, référez vous au 2.1. Et chacune de ces exécutions donne lieux à des tâches différentes.

3.3 `Main.java`

Ce programme comprend vingt fonctionnalités. Dans tous les cas, il est demandé à l'utilisateur d'entrer la taille du buffer.
Le numéro en début de ligne indique quel entier entrer pour choisir la tâche à exécuter :

0. *Créer un fichier rempli aléatoirement*
   Le programme demande à l'utilisateur les caractéristiques de la séquence aléatoire à créer : le nombre d'entrées, le nombre de données pour chaque entrée, et le nom du fichier dans lequel

stocker la séquence. Il ne faut pas préciser un chemin, mais seulement un nom de fichier non existant dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`. Ce fichier est ensuite créé directement dans ce répertoire.
Cette fonctionnalité est utile au cas où l'utilisateur ne dispose pas de Wiimote.

1. *Lire un fichier*
   Le programme demande à l'utilisateur le nom du fichier stockant la séquence à afficher. L'utilisateur peut entrer le chemin vers ce fichier à partir de `Stage_final/Project/Datas/Gestes/`. Les données sont ensuite affichées dans le shell.

2. *Afficher graphiquement une séquence à partir d'un fichier*
   Le programme demande à l'utilisateur le nom du fichier stockant la séquence à afficher. L'utilisateur peut entrer le chemin vers ce fichier à partir de `Stage_final/Project/Datas/Gestes/`. Les données sont ensuite affichées graphiquement.

3. *Ecrire l'interpolation d'une séquence à partir d'un fichier*
   Le programme demande à l'utilisateur le "pas d'interpolation" et le nom du fichier stockant la séquence à interpoler. Ce fichier doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`. Un nouveau fichier nommé "Interpol_" suivi du nom du fichier d'origine est créé dans ce répertoire.

4. *Ecrire l'interpolation d'une base de séquences à partir d'un répertoire*
   Le programme demande à l'utilisateur le "pas d'interpolation" et le nom du répertoire contenant les séquences à interpoler. Ce répertoire doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`. Un nouveau répertoire nommé "Interpol_" suivi du nom du répertoire d'origine, et contenant les séquences interpolées, est créé dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`.

5. *Afficher graphiquement deux séquences à partir de deux fichiers*
   Le programme demande à l'utilisateur le nom de chacun des deux fichiers à afficher. L'utilisateur peut entrer les chemins vers ces fichiers, s'ils se trouvent dans des sous-répertoires du répertoire `Stage_final/Project/Datas/Gestes/`. Cependant, les deux séquences doivent être de même longueur, et avoir le même nombre d'entrées. Les deux séquences sont ensuite affichées graphiquement.

6. *Afficher les données de la WiiRemote graphiquement et écrire les données d'accélération*
   Tout d'abord ne pas oublier d'activer le périphérique Bluetooth. Le programme demande à l'utilisateur d'entrer le nom du dossier dans lequel sauver toutes les séquences enregistrées. Il ne faut pas préciser un chemin, mais seulement un nom de répertoire non existant dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`. Ce répertoire est ensuite créé directement.
   Le programme demande ensuite à l'utilisateur de presser les touches 1 et 2 (en même temps) de la Wiiremote pour l'activer. Quand les signaux des trois accéléromètres s'affichent graphiquement, le programme est prêt à enregistrer des séquences. Pour commencer l'enregistrement, appuyer sur la touche **B**, pareil pour l'arrêter. Pour déconnecter et éteindre la manette, appuyer longuement sur la touche power jusqu'à ce que les diodes s'éteignent.

7. *Afficher graphiquement une séquence avec l'affichage de 'WRLImpl' à partir d'un fichier* Le programme demande à l'utilisateur le nom du fichier stockant la séquence à afficher. L'utilisateur peut entrer le chemin vers ce fichier, à partir de `Stage_final/Project/Datas/Gestes/`. Les données sont ensuite affichées graphiquement.

8. *Ecrire l'intégration d'une séquence à partir d'un fichier*
   Le programme demande à l'utilisateur le nom du fichier stockant la séquence à intégrer. Ce fichier doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`. Un nouveau fichier nommé "Integr_" suivi du nom du fichier d'origine est créé dans ce répertoire.

9. *Ecrire l'intégration d'une base de séquences à partir d'un répertoire*
   Le programme demande à l'utilisateur le nom du répertoire contenant les séquences à intégrer. Ce répertoire doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`. Un nouveau répertoire nommé "Integr_" suivi du nom du répertoire d'origine, et contenant les séquences integrées, est créé dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`.

10. *Apprendre un modèle de séquences à partir d'une base*
    Le programme demande à l'utilisateur le nom du répertoire contenant la base d'apprentissage pour un geste donné, ainsi que le nom du répertoire qui contiendra les séquences du modèle. Il ne faut pas préciser un chemin, mais seulement un nom de répertoire non existant dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`. Ce répertoire est ensuite créé directement dans le répertoire `Stage_final/Project/Datas/Gestes/Autres/`.
    Il est également demandé à l'utilisateur de choisir le nombre de séquences du modèle.

11. *Afficher graphiquement les séquences d'une base à partir d'un répertoire*
    Le programme demande à l'utilisateur d'entrer le nom du répertoire, se trouvant dans le répertoire `Stage_final/Project/Datas/Gestes`, contenant les séquences à afficher à l'écran et graphiquement.

12. *Traduire une base de séquences en un fichier lisible par* `libsvm`
    Le programme demande à l'utilisateur le nom du répertoire contenant les séquences à traduire et le nom du fichier qui contiendra la "traduction", à partir du répertoire `Stage_final/Project/Datas/Gestes/Autres/`.

13. *Ecrire une base de séquences lissées à partir d'un nom et crée les paths de lecture*
    Le programme demande à l'utilisateur le nom du réperoire contenant la base de séquences à lisser. Ce dossier doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes` (et non dans un sous-répertoire). L'utilisateur doit alors instancier deux paramètres du lissage : "pas de lissage" et "beta". Le programme va lisser toutes les séquences des gestes contenues dans le répertoire et les écrira dans un répertoire de `Stage_final/Project/Datas/Gestes` avec un préfice du type `Liss_<pas_de_lissage>_<beta>_`. Pour chaque base du répertoire de destination, un fichier sera écrit dans `Stage_final/Project/java/bin/bases_paths` contenant le chemin de des gestes de la base.

14. *Ecrire une base de séquences dans un fichier lisible par le modèle CrfBranch* Le programme demande à l'utilisateur le nom du répertoire contenant le nom de la base à traiter dans `Stage_final/Project/Datas/Gestes`. L'utilisateur choit alors une base parmi celles proposées. Puis l'utilisateur choisit dans quel dossier de `Stage_final/Project/Datas/CRFs`, le fichier contenant la base sera écrite.

15. *Ecrire une base de séquences centrées-réduites sur chaque dimension à partir d'un nom et crée les paths de lecture*
    Le programme demande à l'utilisateur le nom du réperoire contenant la base de séquences à centrer-réduire. Ce dossier doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes` (et non dans un sous-répertoire). Le programme va centrer-réduire toutes les séquences par dimension des gestes contenues dans le répertoire avec la boîte `BoxCentrerReduire` et les écrira dans un répertoire de `Stage_final/Project/Datas/Gestes` avec un préfice du type `CR_`. Pour chaque

base du répertoire de destination, un fichier sera écrit dans `Stage_final/Project/java/bin/bases_paths` contenant le chemin de des gestes de la base.

16. *Ecrire une base de séquences réduites sur chaque dimension à partir d'un nom et crée les paths de lecture*
Le programme demande à l'utilisateur le nom du réperoire contenant la base de séquences à réduire. Ce dossier doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes` (et non dans un sous-répertoire). Le programme va réduire par dimension toutes les séquences des gestes contenues dans le répertoire et les écrira dans un répertoire de `Stage_final/Project/Datas/Gestes` avec un préfice du type `R_`. Pour chaque base du répertoire de destination, un fichier sera écrit dans `Stage_final/Project/java/bin/`
`bases_paths` contenant le chemin de des gestes de la base.

17. *Ecrire une base de séquences fenêtrées à partir d'un nom et crée les paths de lecture*
Le programme demande à l'utilisateur le nom du réperoire contenant la base de séquences à fenêtrer. Ce dossier doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes` (et non dans un sous-répertoire). Le programme va fenêtrer toutes les séquences des gestes contenues dans le répertoire et les écrira dans un répertoire de `Stage_final/Project/Datas/Gestes` avec un préfice du type `Win_<taille_fenêtre>_`. Pour chaque base du répertoire de destination, un fichier sera écrit dans `Stage_final/Project/java/bin/bases_paths` contenant le chemin de des gestes de la base.
On remarque qu'une séquence fenêtrée sera plus "courte" que la séquence de départ.

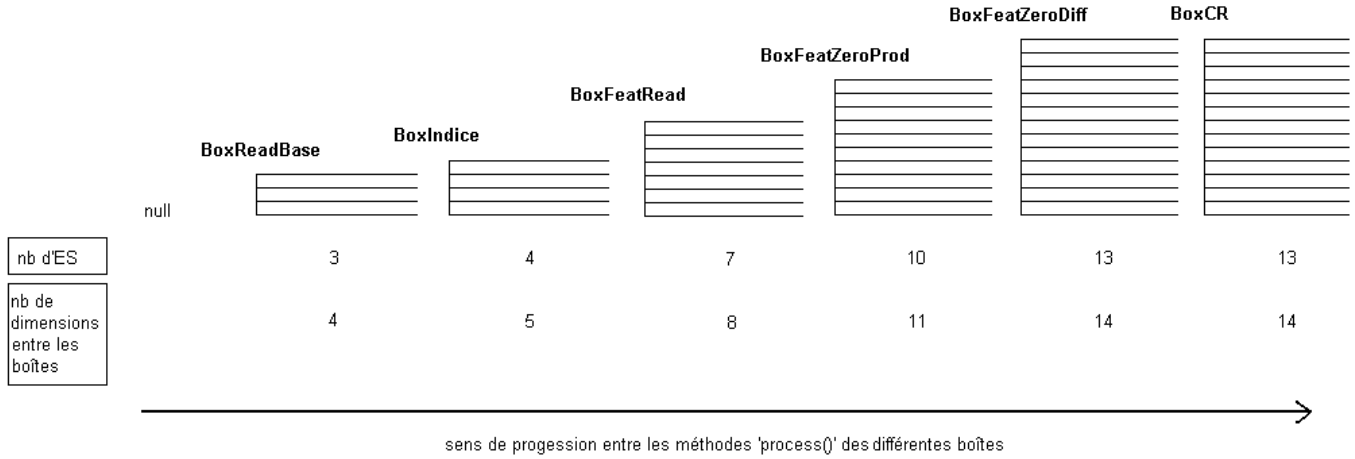
18. *Ecrire une base de séquences avec des features à partir d'un nom et crée les paths de lecture*
Le programme demande à l'utilisateur le nom du réperoire contenant la base de séquences à featurer. Ce dossier doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes` (et non dans un sous-répertoire). Par défaut, les features ajoutés sont résumés par le schéma dans le 3.3.1. Le programme va featurer toutes les séquences des gestes contenues dans le répertoire et les écrira dans un répertoire de `Stage_final/Project/Datas/Gestes` avec un préfice du type `Feat_CR<méthode_centrer_réduire>`. En effet, les séquences sont ici centrées-réduites "par base" à l'aide de la boîte `BoxCR`. Pour chaque base du répertoire de destination, un fichier sera écrit dans `Stage_final/Project/java/bin/bases_paths` contenant le chemin de des gestes de la base.
On remarque qu'il faut d'abord traiter la base avec le cas 19 avant d'utiliser ce cas.

19. *Ecrire une base de séquences lissées avec les features de la boîte Zero à partir d'un nom et crée les paths de lecture*
Le programme demande à l'utilisateur le nom du réperoire contenant la base de séquences à lisser puis featurer. Ce dossier doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes` (et non dans un sous-répertoire). Le programme va lisser puis ajouter le nombre de passage par 0 pour chaque dimension à toutes les séquences des gestes contenues dans le répertoire et les écrira dans un répertoire de `Stage_final/Project/Datas/Gestes` avec un préfice du type `Liss_<pas_de_lissage>_<beta>_Zero_`. Pour chaque base du répertoire de destination, un fichier sera écrit dans `Stage_final/Project/java/bin/bases_paths` contenant le chemin de des gestes de la base.

20. *Ecrire une base de séquences centrées-réduites selon 2 méthodes à partir d'un nom et crée les paths de lecture*
Le programme demande à l'utilisateur le nom du réperoire contenant la base de séquences à centrer-réduire. Ce dossier doit se trouver dans le répertoire `Stage_final/Project/Datas/Gestes` (et non dans un sous-répertoire). Le programme va centrer-réduire par base toutes les séquences des gestes contenues dans le répertoire et les écrira dans un répertoire de `Stage_final/Project/`

`Datas/Gestes` avec un préfice du type `Feat_CR<méthode_centrer_réduire>`. Les séquences sont ici centrées-réduites "par base" à l'aide de la boîte `BoxCR`. Pour chaque base du répertoire de destination, un fichier sera écrit dans `Stage_final/Project/java/bin/bases_paths` contenant le chemin de des gestes de la base.

3.3.1 Exemple d'ajout de Features (cas 18)



3.4 Construction des bases de séquences par Cross Training

Les bases "croisées" de séquences sont créées par le programme `Create_base.java`. Elles servent à établir des performances en variant les apprentissages sur la base de départ. En ce sens, il n'y a pas d'intérêt à créer une base d'apprentissage composée de plus de la moitié des séquences de la base d'origine.

Par exemple, supposons que l'on dispose d'une base `Base` de séquences qui contient 30 séquences pour chaque geste et que l'utilisateur souhaite créer 10 séquences d'apprentissage pour chaque geste. Le programme va alors générer aléatoirement trois bases de 10 séquences pour chaque geste (respectivement différentes pour chaque base). Ces trois bases `Base1`, `Base2`, et `Base3` joueront alors le rôle de base d'apprentissage l'une après l'autre, contre les deux autres.

Considérons un exemple simplifié avec une base de 6 séquences et 2 séquences d'apprentissage pour chaque geste. Le programme va générer aléatoirement trois (=*floor(6/2)*) bases :

Base	Base1	Base2	Base3
seq1, seq2, seq3, seq4, seq5, seq6 seq4, seq5, seq6	seq2, seq4	seq5, seq1	seq6, seq3

Enfin, le programme va générer 6 fichiers lisibles par CrfBranch, du type :

train1_2	seq2, seq4
test1_4	seq1, seq3, seq5, seq6
train2_2	seq5, seq1
test2_4	seq2, seq3, seq4, seq6
train3_2	seq6, seq3
test3_4	seq1, seq2, seq4, seq5

3.5 `TestDTW.java` et `TestCRF.java`

Ces deux programmes permettent de reconnaître des gestes au travers de la Wiimote, à partir de modèles. On se place donc dans un mode supervisé.
Le procédé est le même lorsque l'on exécute l'un de ces deux programmes : le système nous demande d'instancier la variable globale `buffer_length` puis, il faut connecter la Wiimote. Le système reconnait alors les gestes effectués par l'utilisateur. Enfin, le programe se termine par la suppression des fichiers temporaires (séquences de gestes écrites).

- `TestDTW.java` utilise le modèle *Dynamic Time Warping*. Les modèles d'apprentissage sont générés par le programme `Main.java` fonctionnalité 10. Par défaut, ils se trouvent dans le dossier `Stage_final\Project\Datas\ModelesDTW`.
- `TestCRF.java` utilise le modèle *Conditionnal Random Fields Branch* [6]. Les modèles d'apprentissage sont générés par le programme `TrainTestCRFBranch.java`. Ils se trouvent le répertoire du dossier `Stage_final\Project\Datas\CRFs`, indiqué par le paramètre `workspace_local` dans le fichier `Stage_final\Project\java\bin\crfbranch_parametres.txt`.

Vous trouverez des informations plus fournies sur ces deux modèles dans le Manuel du Programmeur[3].

Références

[1] Michael Diamond (aka ChaOs). *WiiRemoteJ library*, 2007. Software available at http://www.wiili.org/index.php/WiiremoteJ.

[2] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM : a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[3] Nicolas Cheifetz. Manuel du programmeur, 2008.

[4] Yasmina Seddik Nicolas Cheifetz and Samuel Ortmans. *Projet IAD - Master 1 IAD*, 2008. Project available at http://che.nico.ifrance.com/PIAD.

[5] Frederic DE STEUR. Wiimotecommander, 2008. Software available at http://sourceforge.net/project/showfiles.php?group_id=222335.

[6] DO Trinh Minh Tri. Multi branch conditional random fields implementation, 2007. http://webia.lip6.fr/~do/pmwiki/index.php/Main/Codes.