HMP-Detector Documentation

Barbara Bruno

February 28, 2014

Contents

Chapter 1

Preliminary Info

1.1 Contact Info

Barbara Bruno (Ph.D. student)
Laboratorium lab
department of Computer Science, Bioengineering, Robotics Engineering, Systems Engineering (DIBRIS)
University of Genova, Italy
Via all'Opera Pia 13, 16145, Genova, Italy
`barbara.bruno@unige.it`

1.2 References

The **docs** folder provided with this distribution contains:

- an introductory PowerPoint presentation providing an overview of the system (ppsx format it requires Microsoft PowerPoint Viewer);
- this user guide.

Detailed information about the system can be found at:

- Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T., Zaccaria, R.: Human motion modelling and recognition: A computational approach. In: IEEE Int Conf on Automation Science and Engineering (CASE), pp. 156–161 (2012)
- Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T., Zaccaria, R.: Analysis of human behavior recognition algorithms based on acceleration data. In: IEEE Int Conf on Robotics and Automation (ICRA), pp. 1602–1607 (2013)

Chapter 2

System Requirements

2.1 OS Requirements

The system has been tested under:

- Ubuntu 11.10:
  free download at `http://releases.ubuntu.com/11.10/`
- Ubuntu 12.04.2 LTS:
  free download at `http://old-releases.ubuntu.com/releases/precise/`

Installation instructions in this guide refer to Ubuntu 12.04.2 OS.

2.2 Hardware Requirements

The HMPdetector allows for both off-line and on-line analysis of accelerometer data for: (i) detection of Human Motion Primitives (HMP); (ii) detection of falls.

The on-line analysis requires the following three devices:

- **collector**, shown in Figure 2.1, to be connected to a USB port of the computer running the HMPdetector software;
- **wrist device**, shown in Figure 2.2, to be properly attached at the user right wrist;
- **waist device**, shown in Figure 2.3, to be properly attached at the user belt.

The wrist device requires to be loaded with the Arduino function `NonClassifier.ino` that can be found inside folder `./arduino/NonClassifier/` inside the HMPdetector package.

The waist device requires to be loaded with the Arduino function `FallDetector.ino` that can be found inside folder `./arduino/FallDetector/` inside the HMPdetector package.

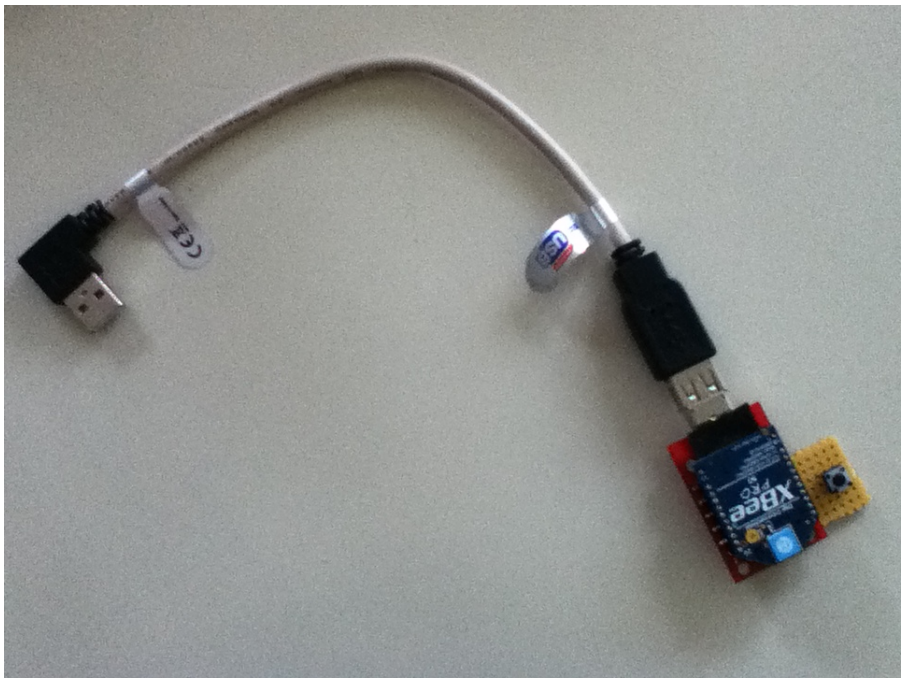

Figure 2.1: Collector device: for sensory data acquisition

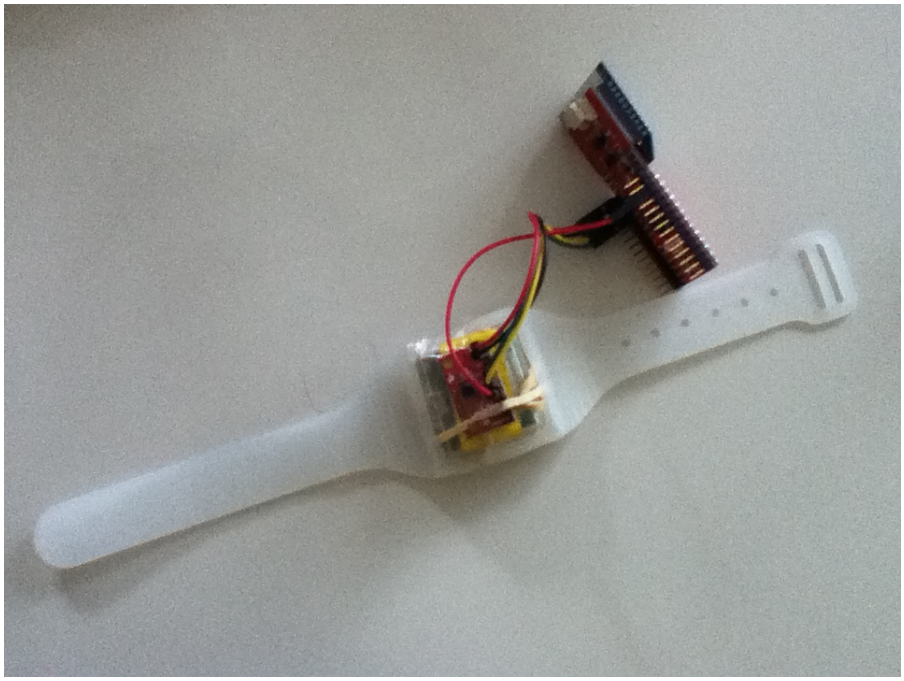

Figure 2.2: Wrist-placed sensing device: for HMP detection

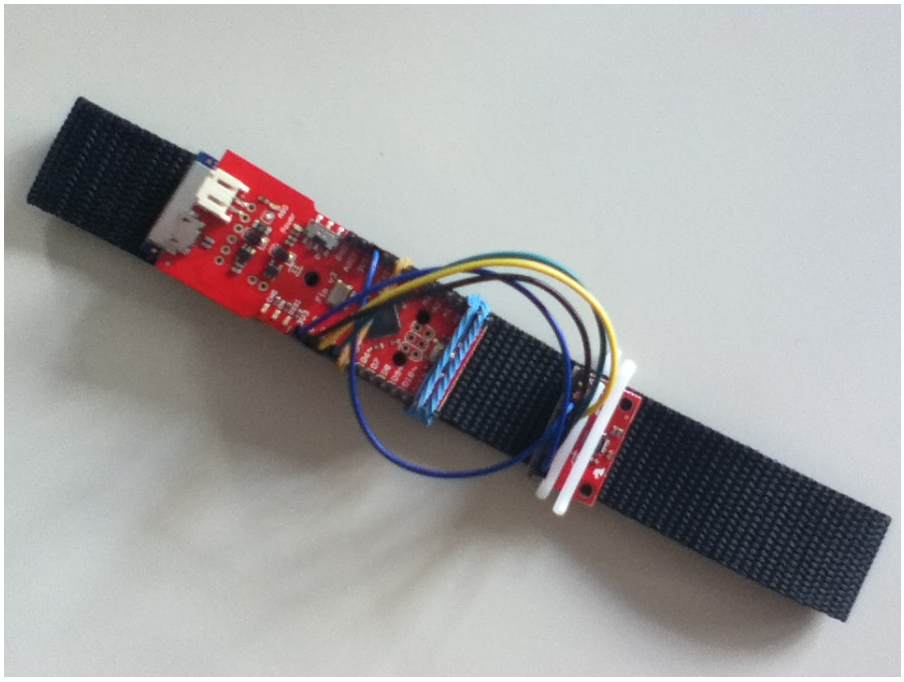

Figure 2.3: Waist-placed sensing device: for Fall detection

Chapter 3

Getting Started

3.1 Package description

The HMPdetector package is provided with the following subfolders:

Name	Type	Description
arduino	folder	Arduino middleware embedded on the sensing devices
docs	folder	HMPdetector documentation
libs	folder	C++ static libraries required by the HMPdetector software
Models	folder	Training datasets for all provided basic Human Motion Primitives
Results	folder	Output folder for all the debug/test functions
Validation	folder	Validation dataset for all provided basic Human Motion Primitives

and the following files:

Name	Type	Description
classifier.cpp classifier.hpp	C++ class	HMPdetector classifier module
CMakeLists.txt	CMake file	CMake build script
creator.cpp creator.hpp	C++ class	HMPdetector models creator module
FallDetector.cpp FallDetector.hpp	C++ class	Fall detector module
HMPdetector.cpp	C++ main	HMPdetector main
reasoner.cpp reasoner.hpp	C++ class	HMPdetector reasoner module
utils.cpp utils.hpp	C++ methods	Frequently used functions
viewPossibilities.m	MATLAB/Octave script	Visualization method for the HMPdetector classifier output

3.2 Installation Guide

3.2.1 Third-party libraries

HMPdetector makes use of: (i) BOOST (collection of portable C++ source libraries), (ii) ARMADILLO (C++ linear algebra library), which requires LAPACK and BLAS libraries for optimal performance, and (iii) PEIS (middleware for Ambient Intelligence applications).

Detailed information about BOOST can be found at:
`http://www.boost.org/`

Detailed information about ARMADILLO can be found at:
`http://arma.sourceforge.net/`

Detailed information about PEIS can be found at:
`http://aass.oru.se/~peis/frameset_page.html`

The HMPdetector package comes with MATLAB/Octave scripts for graphical display of the system output. The download and installation instructions for both MATLAB (licensed) and Octave (free) are NOT included in this Installation Guide.

Detailed information about Octave can be found at:
`http://www.gnu.org/software/octave/`

3.2.2 Installation Procedure

1. With Ubuntu/Synaptic Package Manager download and install the following packages (if not already present in the system):
   - cmake (2.8.5-1ubuntu1 / 2.8.7-0ubuntu5)
   - libblas-dev (1.2.20110419 / 1.2.20110419-2ubuntu1)
   - liblapack-dev (3.3.1-1)
   - zlib1g-dev (1:1.2.3.4.dfsg-3ubuntu4)
   - libglib2.0-dev (2.32.4-0ubuntu1)
   - libglade2-dev (1:2.6.4-1ubuntu1.1)
2. Install the set of basic packages for installation/compilation:
   (a) open a Terminal tab
   (b) type `sudo apt-get install build-essential checkinstall` (insert user password when prompted)
3. Download Boost library (1.46.1) - free download at:
   `http://sourceforge.net/projects/boost/files/boost/1.46.1/`
4. Install Boost library:
   (a) **please refer to the Getting Started Guide in the index.html file inside the package**
   (b) extract the package
   (c) in a Terminal tab move to the folder containing the extraction folder of the Boost package

(d) type `sudo mv boost_1_46_1 /usr/local`
(insert user password when prompted)

(e) re-extract the package

(f) in a Terminal tab move to the new extraction folder of the Boost package

(g) type
`./bootstrap.sh --with-libraries=thread,system,date_time --prefix=/usr/local`

(h) type `sudo ./bjam install`
(insert user password when prompted)

5. Download Armadillo library (3.920.2) - free download at:
`http://arma.sourceforge.net/download.html`

6. Install Armadillo library:

(a) **please refer to the Installation Guide in the README.txt file inside the package**

(b) move inside the extraction folder of the Armadillo package

(c) open the file CMakeLists.txt

(d) change `set(ARMA_USE_LAPACK false)`
to `set(ARMA_USE_LAPACK true)`

(e) change `set(ARMA_USE_BLAS false)`
to `set(ARMA_USE_BLAS true)`

(f) save and close

(g) in a Terminal tab move inside the extraction folder of the Armadillo package

(h) type `cmake .`

(i) type `make`

(j) type `sudo make install`
(insert user password when prompted)

7. Download PEIS middleware (0.6.0.0) - free download at:
`ftp://aass.oru.se/hidden/saffiotti/RobotEraTutorial_120416.tgz`

8. Install PEISkernel (of PEIS middleware):

(a) **please refer to the Installation Guide in the offline-tutorial.pdf file inside the package**

(b) open a Terminal tab

(c) type `sudo apt-get install automake`
(insert user password when prompted)

(d) type `sudo apt-get install libtool`
(insert user password when prompted)

(e) move to folder Software/peiskernel/G6 inside the extraction folder of the PEIS package

(f) type `./autogen.sh`

(g) type `./configure`

(h) type `make`

(i) type `sudo make install`
(insert user password when prompted)

9. Install TupleView (of PEIS middleware):

   (a) **please refer to the Installation Guide in the offline-tutorial.pdf file inside the package**

   (b) in a Terminal tab move to folder Software/tupleview/G6 inside the extraction folder of PEIS package

   (c) type `./autogen.sh`

   (d) type `./configure`

   (e) type `make`

   (f) type `sudo make install`
   (insert user password when prompted)

10. Build the HMPdetector:

    (a) in a Terminal tab move to the extraction folder of the HMPdetector package

    (b) type `cmake .`

    (c) type `make`

3.3 Hardware Setup

TO BE DONE!

3.4 System Overview

HMPdetector provides a command-line interface for the Models Creator, Classifier, Fall detector and Reasoner modules. More specifically:

- the **Models Creator** module builds the models of the considered human motion primitives by applying Gaussian Mixture Modelling and Gaussian Mixture Regression over a provided set of modelling trials (stored inside the motion-corresponding folder inside the Models folder);

- the **Classifier** module performs (i) off-line analysis of recorded acceleration data (stored inside the Validation folder) and (ii) on-line analysis of acceleration data coming from the wrist device. It returns the possibility value of each motion primitive to be the one executed at each time instant;

- the **Fall detector** module performs on-line acquisition of the fall alarm and related signals generated by the waist device in case a fall is detected and integrates them with wrist-motion information extracted from the acceleration data coming from the wrist device;

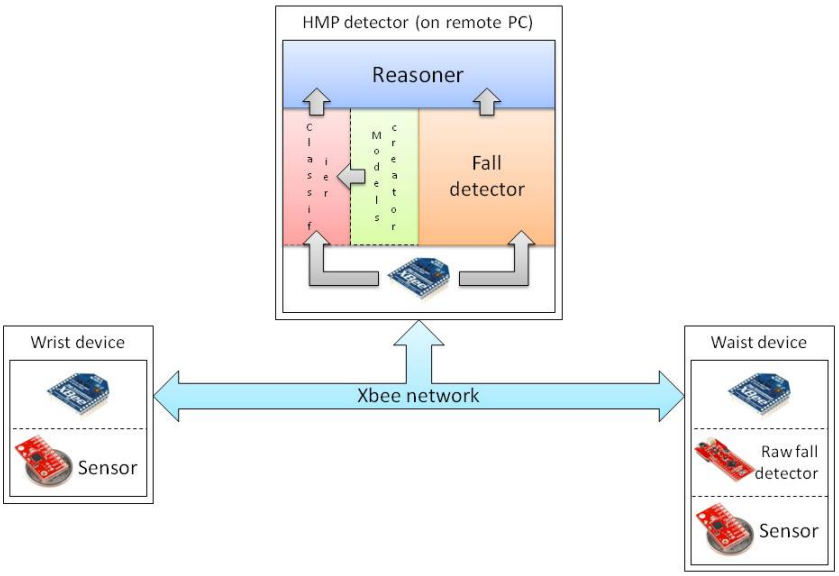

Figure 3.1: System architecture

- the **Reasoner** module performs (i) off-line and (ii) on-line analysis of the possibility values computed by the Classifier module and publishes the results as tuples in the PEIS tuple space under the key activity. It publishes the results of the fall detection performed by the Fall detector module as tuples under the key fall.

To access function-specific basic information and usage examples within the system, in a Terminal tab move to the HMP Detector extraction folder and type `./HMPdetector -h` or `./HMPdetector--help`.

3.5 Basic Human Motion Primitives

HMPdetector comes with the training and validation dataset for 9 basic Human Motion Primitives:

Name	Description	Characteristics
Climb	to climb 3 steps of a staircase	#simple, #recursive, #full-body
GetUp	to get up from a lying position on a bed	#simple, #full-body
PickUpDrink	to pick up a glass from the table and start drinking from it	#complex, #hand-only
PickUpPour	to pick up a bottle from the table and start pouring its content in a glass on the table	#complex, #hand-only
PutDownDrink	to stop drinking from a glass and put it down on the table	#complex, #hand-only
PutDownDrink	to stop pouring the content of a bottle in a glass on the table and put it down on the table	#complex, #hand-only
Sit	to sit down on a chair	#simple, #full-body
Stand	to stand up from a chair	#simple, #full-body
Walk	to take 3 steps	#simple, #recursive, #full-body

Chapter 4

Tutorials

This and following tutorials assume that the HMPdetector executable file is located in folder: `Documents/HMPdetector`.

This and following tutorials assume that the HMPdetector has been correctly installed and built, according to the procedure detailed in Section 3.2.

4.1 Display program help

1. open a new Terminal tab
2. move inside the HMPdetector folder
   (type `cd Documents/HMPdetector`)
3. type `./HMPdetector --help`
   (or equivalently `./HMPdetector -h`)

4.2 Create the models of the default modelling dataset

Assumptions:

- the default modelling dataset is stored in folder: `./Models/Ovada`.

Procedure:

1. open a new Terminal tab
2. move inside the HMPdetector folder
   (type `cd Documents/HMPdetector`)
3. type `./HMPdetector --model`
   (or equivalently `./HMPdetector -m`)

4.3 Create the models of an existing modelling dataset

Assumptions:

- there exists a valid modelling dataset in folder: `./Models/myDataset`.

Procedure:

1. open a new Terminal tab
2. move inside the HMPdetector folder
   (type `cd Documents/HMPdetector`)
3. type `./HMPdetector --model myDataset`
   (or equivalently `./HMPdetector -m myDataset`)

4.4 Load a set of existing models with default loading options

Assumptions:

- there exists a valid modelling dataset in folder: `./Models/myDataset`;
- the corresponding models have been created as detailed in Tutorial 4.3.

Procedure:

1. open a new Terminal tab
2. move inside the HMPdetector folder
   (type `cd Documents/HMPdetector`)
3. type `./HMPdetector --load myDataset`
   (or equivalently `./HMPdetector -l myDataset`)

4.5 Validate a loaded model with an existing set of validation trials

Assumptions:

- there exists a valid modelling dataset in folder: `./Models/myDataset`;
- the corresponding models have been created as detailed in Tutorial 4.3;
- the corresponding models have been loaded as detailed in Tutorial 4.4;
- there exists a valid validation dataset composed of `[N]` trials for the model `[myHMP]` in folder: `./Validation/myDataset`.

Procedure:

1. open a new Terminal tab

2. move inside the HMPdetector folder
   (type `cd Documents/HMPdetector`)

3. type `./HMPdetector --validate [myHMP] myDataset [N]`
   (or equivalently `./HMPdetector -v [myHMP] myDataset [N]`)

4.6 Perform off-line classification of an existing long recording

Assumptions:

- there exists a valid modelling dataset in folder: `./Models/myDataset`;
- the corresponding models have been created as detailed in Tutorial 4.3;
- the corresponding models have been loaded as detailed in Tutorial 4.4;
- there exists a valid recording `mylongTest.txt` in folder: `./Validation/longTest`.

Procedure:

1. open a new Terminal tab

2. move inside the HMPdetector folder
   (type `cd Documents/HMPdetector`)

3. type `./HMPdetector --test mylongTest.txt`
   (or equivalently `./HMPdetector -t mylongTest.txt`)

Chapter 5

Advanced Operations

5.1 Modify the modelling parameters of an existing modelling dataset

Assumptions:

- there exists a valid modelling dataset in folder: `./Models/myDataset`.

A valid modelling dataset contains a configuration file named `HMPconfig.txt`, which defines:

- the HMP to be modelled, among the ones provided in the dataset;
- the modelling parameters for each considered HMP.

Each line in the file `HMPconfig.txt` corresponds to one HMP to be modelled by the creator and is defined according to the following convention:

[name] [nbMT] [nbGG] [nbBG]

where:

- **[name]** is the name of the HMP to be modelled by the creator, matching the name of the folder inside the dataset containing the modelling trials for the HMP;
- **[nbMT]** is the number of modelling trials for the HMP;
- **[nbGG]** is the number of Gaussian clusters that should be used by the creator to model the gravity feature of the HMP;
- **[nbBG]** is the number of Gaussian clusters that should be used by the creator to model the body acceleration feature of the HMP.

By removing a line from the file `HMPconfig.txt` the corresponding HMP will not be modelled by the creator.

By adding/modifying a line of the file `HMPconfig.txt` the corresponding HMP will be modelled by the creator according to the specified parameters.

5.2 Modify the classification parameters of an existing modelling dataset

Assumptions:

- there exists a valid modelling dataset in folder: `./Models/myDataset`.

A valid modelling dataset contains a configuration file named `Classifierconfig.txt`, which defines:

- the HMP to be considered for classification, among the ones provided in the dataset;
- the classification parameters for each considered HMP.

The first line in the file `Classifierconfig.txt` defines the total number of HMP to be considered for classification.

Each subsequent line corresponds to one HMP to be considered by the classifier and is defined according to the following convention:

[name] [wG] [wB] [th]

where:

- **[name]** is the name of the HMP to be considered by the classifier, matching the name of the folder inside the dataset containing the modelling trials for the HMP;
- **[wG]** is the weight of the gravity feature for the HMP (default: 0.5);
- **[wB]** is the weight of the body acceleration feature for the HMP (default: 0.5);
- **[th]** is the threshold on the distance to the model for a positive recognition of the HMP.

By removing a line from the file `Classifierconfig.txt` the corresponding HMP will not be considered by the classifier.

By adding/modifying a line of the file `Classifierconfig.txt` the corresponding HMP will be considered by the classifier and classification will be performed according to the specified parameters.