Lehrstuhl für Medieninformatik
Institut für Information und Medien, Sprache und Kultur
Universität Regensburg

Time-Series Data, PyQtGraph

Interaction Techniques and Technologies (ITT), SS 2016

Session 11 (09.06.2016), Raphael Wimmer

Overview

These are slides/notes for the lecture, automatically generated from the slide set. Please extend this outline with your own notes.

Goals for this Week

Overall: Understanding the basics of digital signal processing

Know

- time-series data
- noise, filters
- filter kernels
- Fourier transform

Learn

- implementing signal processing chains (with PyQtGraph)
- how to choose and implement appropriate filters for certain tasks

Practice

- Python
- Analyzing and filtering signals with numpy (and IPython notebook)

Today

- **14:20 - 14:40** Overview, Wiimote
- **14:40 - 15:00** Experiment: bubble level
- **15:05 - 15:25** Introduction: Time-series data, digital signals
- **15:25 - 15:45** Overview: PyQtGraph

Where are We?

- **Conducting and Logging Experiments** (+ intro to Python / PyQT)
- **Documenting and Visualizing Experiments** (+ intro to pylab, matplotlib)
- **Pointing** (pointing devices, Fitts' Law, Steering Law, CD gain, …)
- **Text Entry** (speed, models, keyboard layouts, input techniques)
- **Models of Interaction** (KLM, GOMS)
- **Gestural Input, Sensors** (WiiMote, intro to PyQtGraph)

New Groups

- Find a new group member, tell me about it.

The Wiimote

Nintendo Wii Remote ('Wiimote')

Figure 1: source:https://en.wikipedia.org/wiki/File:Wii_Remote_Image.jpg

Hardware

- 11 buttons
- 4 LEDs
- Rumble motor
- lo-fi speaker
- 3-axis accelerometer
- 3-axis gyroscope (Wii MotionPlus)
- point-tracking IR camera
- extension port using I2C protocol
- communication via custom Bluetooth HID protocol
- comprehensive documentation at WiiBrew[1]

[1] http://wiibrew.org/wiki/Wiimote

Connecting to the Wiimote

- protocol reverse-engineered, documented at WiiBrew[2]
- numerous libraries supporting the Wiimote (List at WiiBrew[3])
- very few (e.g., WiiRemoteJ[4]) support all Wiimote features
- most libraries/bindings for Python either abandoned, incomplete, and/or missing support for current *Wii Remote Plus*
- -> own implementation: `wiimote.py`

wiimote.py

- pure Python implementation using PyBluez (Windows, Linux, (OS X))
- usage:
  - `import wiimote`
  - `addr, name = wiimote.find()`
  - `wm = wiimote.connect(addr)`
- working in v0.3:
  - LEDs: `wm.leds[2] = True`
  - Rumble motor: `wm.rumble()`
  - Buttons: `print(wm.buttons["A"])`
  - Accelerometer: `print(wm.accelerometer)`
  - Infrared camera: `print(wm.ir)`
  - Polling or callbacks for buttons, accelerometer, infrared camera
- issues in v0.3:
  - old Wiimotes not supported
  - code not fully PEP8-compliant and missing docstrings
  - Gyroscope and loudspeaker support not implemented yet
- download from GitHub: https://github.com/RaphaelWimmer/wiimote.py

Quickstart

- install pybluez via PyPI: `pip3 install pybluez`
- get a Wiimote
- write down its Bluetooth MAC address (see label on side)
- press the `sync` button on the back of the Wiimote.
- Discover and 'Trust' the Wiimote using your distribution's Bluetooth manager
- download wiimote.py
- run `python3 wiimote_demo.py <bt_addr>` for a quick demo, adjust the code to your liking.

Tips for getting Bluetooth running in a virtual machine

- Bluetooth modules that are connected to the computer via USB usually work without problems. If the modules is connected via another bus, it is not possible to use it from within VirtualBox. However, VMWare apparently supports passing through Bluetooth directly.
- See https://help.ubuntu.com/community/VirtualBox/USB for information on how to forward a USB device to a virtual machine.

[2] http://wiibrew.org/wiki/Wiimote
[3] http://wiibrew.org/wiki/Wiimote/Library
[4] https://github.com/micromu/WiiRemoteJ

- **OS X as host:** you need to unload the Bluetooth kernel modules first so that the Bluetooth device can be forwarded to the virtual machine. You may use the *Bluetoothtoggler* script in GRIPS for this purpose.

Exercise: Digital Bubble Level

Write a small Python application `level.py` that takes a Bluetooth MAC address as its only parameter. This application should turn your WiiMote into a digital bubble level with the following properties:

- the accelerometer measures inclination in X and Y axis
- the LEDs show direction and amount of deviation from the horizontal: `*---`, `**--`, `-*--`, …
- once the WiiMote is perfectly horizontal, all LEDs light up and the WiiMote rumbles once.
- the directional buttons allow for setting the axis of measurement to be used (X or Y).

The application should `import wiimote` similar to the `wiimote_demo.py` example.

Hint: Activating the rumble motor will mess with the accelerometer values. You might want to wait for a short time until you read and interpret them again.

Time-Series Data

Overview

- analog / digital signals
- sampling rate, Nyquist theorem
- descriptive statistics
-

PyQtGraph: Scientific Graphics and GUI Library for Python

What is PyQtGraph?

- http://pyqtgraph.org/
- interactive 2D and 3D plots
- based on PyQt4 (preliminary pyqt5 support available on GitHub)
- used primarily in scientific computing (example[5])
- flowchart support (talk on flow-based programming in general[6])
- installation: `apt-get install python3-pyqtgraph`
- run examples: `python3 -m pyqtgraph.examples`

Demonstration: Visualize Wiimote Accelerometer Data

- check out the PyQtGraph examples
- check out `wiimote_node.py` and run it

[5] http://journal.frontiersin.org/Journal/10.3389/fninf.2014.00003/full
[6] https://www.youtube.com/watch?v=3oaelUXh7sE/

Outlook

Next Session

- More fun with the Wiimote (IR camera)
- Digital Signal Processing (DSP): filters and transformations

Course Assignment

- Implement a digital water level (continuation of in-course exercise)
- Extend the Wiimote flowchart example
- Read up on DSP

ENDE