Lathund för analys av IMU-data för mörkerstudien

Kjartan Halvorsen

September 27, 2016

Contents

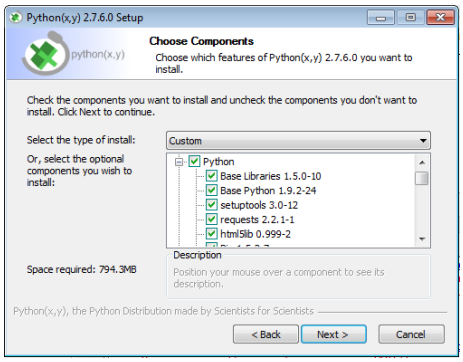
1 Översikt

Data från IMU'erna delas upp och märks med försöksperson nummer och typ av försök (monokulär, binokulär, etc), för att därefter läggas in i en databas på formatet hdf5. Detta det smidigt att plocka fram och bearbeta delar av data. Första steget är därför att förbereda data för att läggas in i databasen. Därefter finns det ett antal olika kommandon för att göra beräkningar på data.

2 Program att installera

2.1 Python

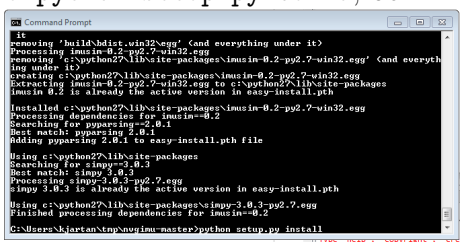
- Python(x,y) scientific python
- Ladda ner installationsprogrammet (drygt 600Mb).
- Öppna installeraren. Acceptera licensen
- I dialogen där du väljer vad som ska installeras, så välj allt genom att klicka i check-boxen framför Python och framför Other



2.2 Python-kod för att bearbeta imudata

- nvgimu zip-fil
- Extrahera filerna till valfri plats på hårddisken.
- Öppna kommandofönstret (*command prompt*)

- Gör `cd` (change directory) till katalogen `nvgimu` som du precis extraherat.
- Installera programmet genom att ange `c:\blabla\nvgimu>python setup.py build`, och därefter `c:\blabla\nvgimu>python setup.py install`.

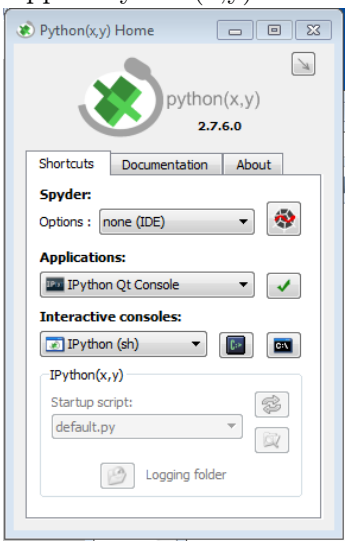


2.3 Tidigare data

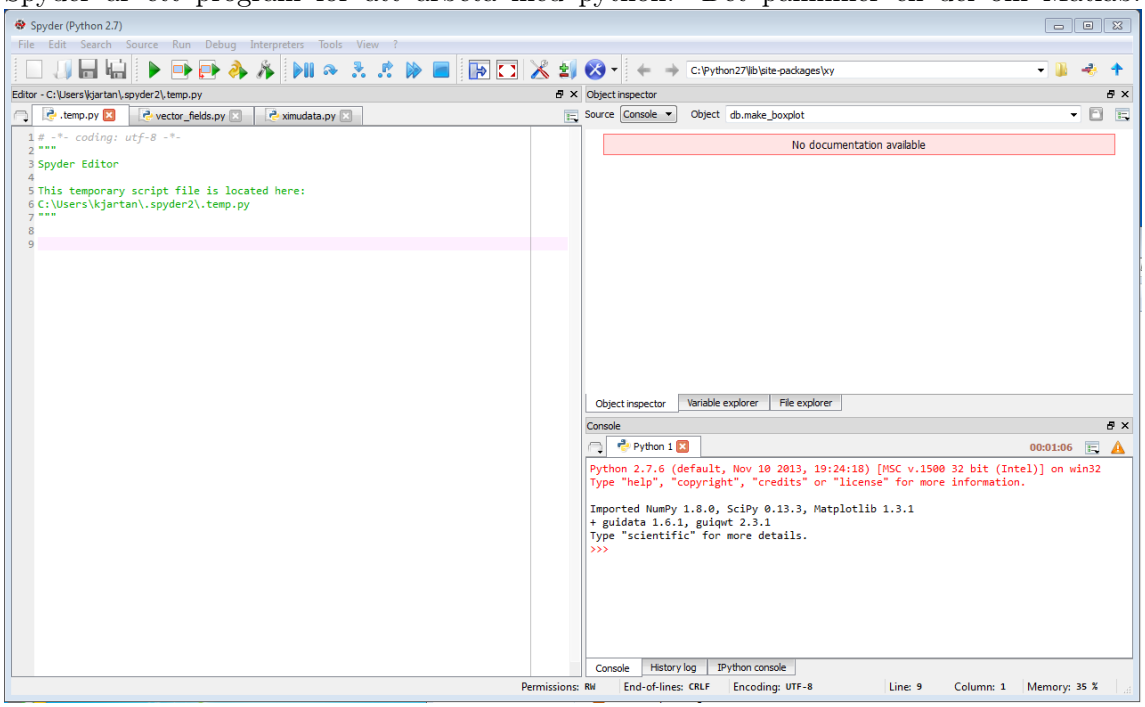
Data från försök i labbet 2012 finns samlade i en stor hdf5-fil. Denna kan laddas ner här: nvg data hdf-format Spara filen på valfri plats på datorn. Notera platsen, då du kommer behöva den sedan.

3 Öppna programmet

- Öppna Python(x,y) start-dialogen. Den finns under Start->Alla program->Python(x,y).



- Starta Spyder
- Spyder är ett program för att arbeta med python. Det påminner en del om Matlab.



4 Förbereda data

- IMU-data måste finnas tillgänglig i csv-format (comma-spaced values). För varje IMU och försöksperson behövs filerna `blabla_CalInertialAndMag.csv` och `blabla_DateTime.csv`.
- För att smidigt kunna dela upp filerna i separata försök, så behövs en lista med tidsstämpel när försöken startade. T.ex. en fil `blabla_events.txt` som ser ut så här

```
2013-11-01 20:31:14, b
```

```
2013-11-01 20:38:14, m
....
```

- I python-filen `imusum/ximu/ximudata.py` (ca rad 1500) finns en funktion `nvg_2012_09_data()` som returnerar all nödvändig information om var datafiler ligger för den studien som gjordes september 2012. En motsvarande funktion behöver göras för nya set med data.

5 Lägga till data i databasen

I `ximudata.py` görs detta i funktionen `split_files_main` på rad 1450. Först måste databasen öppnas. Sedan läggs data in:

```
>>> import imusim.ximu.ximudata as ximudata
>>> db = ximudata.NVGData('/path/to/my/hdf5file.hdf5')
>>> ximudata.split_files_main(db, rawData=nvg_2013_10_data)
```

Det är viss manuell handpåläggning som måste till för att synkroniseringen ska funka: För varje IMU plottas accelerationen i början av mätfilen. Den dubbla synkroniserings-pulsen borde synas väl. Marker in ett tidsintervall som omfattar pulsen. När detta är gjort för alla IMU'er för försökspersonen, så söks synkroniseringspulsen automatiskt i varje måtfil. Resultatet plottas så att man får en visuell kontroll att algoritmen funkar. Se funktionen `sync_signals()`, rad 1260. Enligt dokumentationen för denna:

```
Will look for two narrow peaks more than 100 samples but less than
400 samples apart in the list of signals at the given channel.
The first threshold passing is taken as the sync signal.
The packet number and signal index at the peak are returned in a list.
```

6 Bearbeta data

Börja med att ladda in databasen:

```
>>> import imusim.ximu.ximudata as ximudata
>>> db = ximudata.NVGData('/path/to/my/hdf5file.hdf5')
```

Databas-objektet har ett antal metoder (funktioner) som kan anropas för att räkna på data. För att se en lista på dessa

```
>>> dir(db)
['__doc__',
 '__init__',
 '__module__',
 'add_imu_data',
 'apply_to_all_trials',
 'close',
 'create_nvg_db',
 'descriptive_statistics_decorator',
 'detect_steps',
 'fix_cycle_events',
 'fname',
 'get_PN_at_sync',
 'get_ROM_joint_angle',
```

```
'get_RoM_angle_to_vertical',
'get_angle_between_segments',
'get_angle_to_vertical',
'get_cycle_data',
'get_cycle_frequency',
'get_imu_data',
'get_minmax_angle_to_vertical',
'get_minmax_joint_angle',
'get_orientation',
'get_range_of_motion',
'get_trial_attribute',
'get_vertical_displacement',
'has_trial_attribute',
'hdfFile',
'list_imus',
'make_boxplot',
'normalize_statistics',
'plot_imu_data',
'track_displacement',
'track_orientation']
```

Det första som måste göras är att hitta fotisätningen, så att data kan delas upp i separata gångsteg. Detta görs med metoden `detect_steps()`. Ett smidigt sätt om man vill köra denna metoden (eller någon annan metod) på alla försök är att använda metoden `apply_to_all_trials`:

```
>>> db.apply_to_all_trials(db.detect_steps)
```

För varje försök plottas accelerationen i IMU'en LA (Left Ankle). Kolla på plotten för att bestämma lämplig tröskel. Isätt (start på steg) bestäms som tidpunkt för varje mätpunkt som överstiger tröskeln, givet att det är åtminstone 240 samples sedan förra isättet. Det är alltså en hårdkodad minsta stegduration. Se koden för metoden på rad 302. Resultatet av `detect_steps()` lagras i databasen. Man behöver alltså inte göra om detta steget, med mindre det finns anledning att tro att steg-detektionen kunde göras bättre med annat värde på tröskeln.

De flesta beräkningar på data bygger på att man har hittat början på steget, och kan dela upp mätningen i separata steg. Aktuella metoder är

```
'get_ROM_joint_angle',
'get_RoM_angle_to_vertical',
'get_angle_between_segments',
'get_angle_to_vertical',
'get_cycle_data',
'get_cycle_frequency',
'get_minmax_angle_to_vertical',
'get_minmax_joint_angle',
'get_range_of_motion',
'get_vertical_displacement',
```

Också dess kan anropas på varje försök eller på delmängder av alla försök. T.ex.

```
>>> res = db.apply_to_all_trials(db.get_RoM_angle_to_vertical, {'imu':'N'},
                                          subjlist=['S2', 'S3'], triallist=['b', 'n'])
```

vilket vill beräkna range of motion för vinkeln mot vertikalen för imu N for försöken "b" och "n" för försökspersonerna S2 och S3.