# CORMORAN Measurement Campaign

The CORMORAN measurement campaign is the first known campaign in the WBAN context gathering :

* 3 differents radio technologies (HiKoB, CEA plateform and Beespoon phone)
* Up to 24 radio devices equiped on a single body
* A precise capture of the radio device and body movement using a Vicon motion capture (MOCAP) system.
* A perfect knowledge of the capture environement
* 58 Series with capture or group navigation scenarios

One of the main characteristics of this measurement campaign is the use of a precise motion capture system which allows to get a ground truth position of any radio device which make the radio observable values open to insightful interpretation.

## Motivation for creating a specific tool

In order to exploit the CORMORAN measurement campaign, a dedicated tool has been envisaged. Regarding that one aim of the CORMORAN project is to provide a simulation plateform from the Channel to the MAC Layer, the tool naturally takes place inside the PyLayers plateform.

This specific tool creation has been motivated by the intrinsec complexity of the the measurement campaign. First, no existing tool are able to exploit simultaneously the radio and MOCAP information from the measures.

The co-existence of 3 different radio technologies implies 3 different file formats which have to be interpreted and combined together to be exploitable. However, the motion capture, and the 3 differents radio acces technologies (RAT) operating at different sample rate, which leads to manipulating 4 different time basis. As well, no automatic start-synchronization mechanism was availble between the different technologies which leads to a non systematic time shift between the different basis.

Finally the aim of a measurement campaign is to easily provide valuable and exploitable information for the project members and more generally by people in the research community. The goal of such a tool is to help and simplify dissemination.

## Prerequisite Installations

Before starting using this tool, some requirements have to be satisfied.

1. The open source platform PyLayers ( http://www.pylayers.org ) has to be installed following the installation notes here: https://github.com/pylayers/pylayers/blob/master/INSTALL.txt
2. The CORMORAN measurements have to be downloaded from the public repository
3. An environement variable $CORMORAN has to be set at the root of your CORMORAN measurements directory (help about setup of environement variables can be found in pylayers ' INSTALL.txt

Once those 3 steps are satisfied, the CORMORAN exploitation measure tool is ready to be used.

# The CorSer Class

The exploitation of measures tool takes place as a specific class named CorSer (which stands for Cormoran Series). Once PyLayers has been installed, it is possible to directly access to the class by importing it.

>>> from pylayers.measures.cormoran import \*.

### Get information on the Series

Before creating the CorSer object it is possible to consult the available measurements series using *cor\_log()*. Then for each **serie** of a given **day** it is possible to get:

* The involved subject(s)
* The radio technology
* A short description of the serie

>>> cor\_log()  
 serie day Subject techno \  
0 1 11 Bernard TCR  
1 2 11 Bernard TCR  
2 3 11 Bernard TCR  
3 4 11 Bernard TCR  
4 5 11 Nicolas HKB+BS  
5 6 11 Nicolas HKB+BS  
6 7 11 Nicolas HKB+BS  
7 8 11 Nicolas HKB+BS  
8 9 11 Bernard TCR  
9 10 11 Bernard TCR  
10 11 11 Bernard TCR  
11 12 11 Bernard TCR  
12 13 11 Nicolas HKB+BS  
13 14 11 Nicolas HKB+BS  
14 15 11 Nicolas HKB+BS  
15 16 11 Nicolas HKB+BS  
16 17 11 Bernard TCR  
17 18 11 Bernard TCR  
18 19 11 Bernard TCR  
19 20 11 Bernard TCR  
20 21 11 Nicolas HKB+BS  
21 22 11 Nicolas HKB+BS  
22 23 11 Nicolas HKB+BS  
23 24 11 Nicolas HKB+BS  
24 25 11 Bernard TCR  
25 26 11 Bernard TCR  
26 27 11 Nicolas HKB+BS  
27 28 11 Nicolas HKB+BS  
28 29 11 Nicolas HKB+BS  
29 30 11 Nicolas HKB+BS  
30 31 11 Nicolas HKB+BS  
31 32 11 Nicolas TCR+HKB+BS  
32 33 11 Nicolas TCR+HKB+BS  
33 34 11 Nicolas TCR+HKB+BS  
34 35 11 Nicolas TCR+HKB+BS  
35 1 12 Nicolas Jihad Eric TCR  
36 2 12 Nicolas Jihad Eric TCR  
37 3 12 Nicolas jihad Eric TCR  
38 4 12 Nicolas Jihad Eric TCR  
39 5 12 Nicolas Jihad Eric TCR  
40 6 12 Nicolas Jihad Eric TCR  
41 7 12 Nicolas Jihad Eric TCR  
42 8 12 Nicolas Jihad Eric TCR  
43 9 12 Nicolas Jihad Eric TCR+HKB+BS  
44 10 12 Nicolas Jihad Eric TCR+HKB+BS  
45 11 12 Nicolas Jihad Eric TCR+HKB+BS  
46 12 12 Nicolas Jihad Eric TCR+HKB+BS  
47 13 12 Nicolas Jihad Eric TCR+HKB+BS  
48 14 12 Nicolas Jihad Eric TCR+HKB+BS  
49 15 12 Nicolas Jihad Eric TCR+HKB+BS  
50 16 12 Nicolas Jihad Eric TCR+HKB+BS  
51 17 12 Nicolas Jihad Eric HKB+BS  
52 18 12 Nicolas Jihad Eric HKB+BS  
53 19 12 Nicolas Jihad Eric HKB+BS  
54 20 12 Nicolas Jihad Eric HKB+BS  
55 21 12 Nicolas Jihad Eric HKB+BS  
56 22 12 Nicolas Jihad Eric HKB+BS  
57 23 12 Nicolas Jihad Eric HKB+BS  
58 24 12 Nicolas Jihad Eric HKB+BS  
  
 Short Notes  
0 Subject Walk circularly  
1 Subject Walk circularly  
2 Subject Walk circularly  
3 Subject Walk circularly  
4 Subject Walk circularly  
5 Subject Walk circularly  
6 Subject Walk circularly  
7 Subject Walk circularly  
8 INTERRUPTED Subject Walk circularly ++ speed  
9 Subject Walk circularly ++ speed  
10 Subject Walk circularly ++ speed  
11 Subject Walk circularly ++ speed  
12 Subject Walk circularly without looking BS pho...  
13 Subject Walk circularly + Navigation movement  
14 Subject Walk slowly without looking BS phone h...  
15 Subject Walk slowly without looking BS phone h...  
16 Static subject pointing corners then yoga post...  
17 Static subject pointing corners then yoga post...  
18 Static subject pointing corners then yoga post...  
19 Static subject pointing corners then yoga post...  
20 Static subject pointing corners (withphone) th...  
21 Static subject pointing corners (withphone) th...  
22 INTERRUPTED Static subject pointing corners (w...  
23 Static subject pointing corners (withphone) th...  
24 Kung-fu Kata  
25 Kung-fu Kata with lost sensor  
26 subject open door, sit, type on leyboard, take...  
27 subject open door, sit, type on leyboard, take...  
28 Crossfade Yoga Posture with phone BS left hand  
29 Crossfade SLOW Yoga Posture with phone BS lef...  
30 Subject Walk circularly  
31 3 turns circularly inc. speed sequentially  
32 3 turns circularly inc. speed sequentially  
33 3 turns circularly inc. Speed + muscle-buildi...  
34 3 turns circularly inc. Speed + muscle-buildi...  
35 DATA ISSUE 3 FireMen Nav  
36 3 FireMen Nav (possible mocap issue)  
37 3 FireMen Nav (possible mocap issue)  
38 INTERRUPTED 3 FireMen Nav  
39 subjects Random walk + new interfering subject...  
40 subjects Random walk + new interfering subject...  
41 subjects slow Random walk + interfering subjec...  
42 subjects slow Random walk + interfering subjec...  
43 SubjectSlow motion: Indoor Nav then Firemen t...  
44 SubjectSlow motion: Indoor Nav then Firemen t...  
45 Subjectnormal speed: Indoor Nav then Firemen ...  
46 Subjectnormal speed: Indoor Nav then Firemen ...  
47 subjects Random walk + new interfering subject...  
48 subjects Random walk + new interfering subject...  
49 subjects Random walk + new interfering subject...  
50 subjects Random walk + new interfering subject...  
51 NO HKB Subjectnormal speed: Indoor Nav then F...  
52 NO HKB Subjectnormal speed: Indoor Nav then F...  
53 NO HKB Subjectnormal speed: Indoor Nav then F...  
54 NO HKB Subjectnormal speed: Indoor Nav then F...  
55 subjects Random walk + new interfering subject...  
56 subjects Random walk + new interfering subject...  
57 subjects Random walk + new interfering subject...  
58 subjects Random walk + new interfering subject...

## Loading a Serie

As an example, serie 6 from day 11 can be loaded using the following command:

>>> S=CorSer(serie=6,day=11)  
  
load infrastructure node position: \*\*\*\* Processor coding : Intel-PC  
  
load Nicolas body: \*\*\*\* Processor coding : Intel-PC  
  
BS data frame index: Align on mocap OK... WARNING time-offset NOT applied  
No BS offset not yet set => use self.offset\_setter  
  
HKB data frame index: Align on mocap OK... time-offset applied OK  
  
Create distance Dataframe... OK

Once loaded information about the serie (date, type, ...) can be obtained just by calling the object itself:

>>> S  
Filename: Sc20\_S6\_R2\_HKBS  
Day : 11/06/2014  
Serie : 6  
Scenario : 20  
Run : 2  
Type : HKBS  
Original Video Id : Single  
Subject(s) : Nicolas  
  
Body available: True  
  
BeSPoon : Sc20\_S6\_R2\_HKBS.csv  
HIKOB : Sc2\_0\_S6\_r2\_HKB\_Single.mat

### Radio DataFrames

Data frames are *Pandas* objects which can be interpreted as tables.

* Each line correspond a given timestamp
* Each column correspond to a given link between 2 radio devices

Depending on available RAT involved in the serie, different data frames are available:

* HiKoB (HKB) data : *S.hkb*
* BeSpoon data : *S.bespo*
* TCR data : *S.tcr*

In the example serie chosen, only HiKoB and Bespoon are available.

Here is an example of the RSS values obtained by the HKB sensors for the 120 available links and the 5 first available timestamp :

>>> S.hkb.head(5)  
 AP1-AP2 AP1-AP3 AP1-AP4 AP1-HeadRight AP1-TorsoTopRight \  
0.000000 NaN NaN NaN NaN NaN  
0.010001 NaN NaN NaN NaN NaN  
0.020002 -60 -64 -61 -71 -81  
0.030003 -60 -64 -61 -71 -81  
0.040004 -60 -64 -61 -71 -81  
  
 AP1-TorsoTopLeft AP1-BackCenter AP1-ElbowRight AP1-ElbowLeft \  
0.000000 NaN NaN NaN NaN  
0.010001 NaN NaN NaN NaN  
0.020002 -73 -78 -79 -84  
0.030003 -73 -78 -79 -84  
0.040004 -73 -78 -79 -84  
  
 AP1-HipRight ... WristRight-WristLeft \  
0.000000 NaN ... NaN  
0.010001 NaN ... NaN  
0.020002 -73 ... -64  
0.030003 -73 ... -64  
0.040004 -73 ... -64  
  
 WristRight-KneeLeft WristRight-AnkleLeft WristRight-AnkleRight \  
0.000000 NaN NaN NaN  
0.010001 NaN NaN NaN  
0.020002 -88 -64 -55  
0.030003 -88 -64 -55  
0.040004 -88 -64 -55  
  
 WristLeft-KneeLeft WristLeft-AnkleLeft WristLeft-AnkleRight \  
0.000000 NaN NaN NaN  
0.010001 NaN NaN NaN  
0.020002 -63 -61 -77  
0.030003 -63 -61 -77  
0.040004 -63 -61 -77  
  
 KneeLeft-AnkleLeft KneeLeft-AnkleRight AnkleLeft-AnkleRight  
0.000000 NaN NaN NaN  
0.010001 NaN NaN NaN  
0.020002 -60 -84 -79  
0.030003 -60 -84 -79  
0.040004 -60 -84 -79  
  
[5 rows x 120 columns]

### Non Radio DataFrames

Extra data frames are also available to acces to non radio information. In particular, it exists :

* *S.devdf*: the device dataframe, which gives mechanical information: position (x,y,z), velocity (v,vx,vy,vz) and acceleration (a,ax,ay,az) of the devices at any time stamps
* *S.distdf*: the distance data frame, which gives ground truth distances between the different radio links.

Here is the 5 last data of the device data frame...

>>> S.devdf.tail(5)  
 id subject x y z v vx \  
104.2 HKB:14 Nicolas 0.158588 -1.574102 0.526740 0.012375 -0.005046  
104.2 HKB:1 0.018552 -2.749937 0.979166 0.000000 0.000000  
104.2 HKB:16 Nicolas -0.229677 -1.445404 0.175125 0.010563 -0.007414  
104.2 HKB:10 Nicolas 0.262695 -1.433168 1.143153 0.057829 -0.048329  
104.2 HKB:3 0.021135 3.375590 1.003871 0.000000 0.000000  
  
 vy vz a ax ay az  
104.2 0.010521 0.004119 2.241849 1.972888 0.738384 0.767065  
104.2 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000  
104.2 -0.006640 -0.003540 0.547761 0.122199 -0.250196 -0.471711  
104.2 -0.030039 -0.010302 0.924303 -0.697193 0.368582 -0.482085  
104.2 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

... and the 5 last data of the distance data frame:

>>> S.distdf.tail(5)  
 HKB:1-HKB:2 HKB:1-HKB:3 HKB:1-HKB:4 HKB:1-HKB:5 HKB:1-HKB:6 \  
104.159996 6.102589 6.125578 6.135849 1.308815 1.163639  
104.169997 6.102589 6.125578 6.135849 1.309074 1.163713  
104.179998 6.102589 6.125578 6.135849 1.309470 1.163938  
104.189999 6.102589 6.125578 6.135849 1.309873 1.164064  
104.200000 6.102589 6.125578 6.135849 1.310357 1.164289  
  
 HKB:1-HKB:7 HKB:1-HKB:8 HKB:1-HKB:9 HKB:1-HKB:10 HKB:1-HKB:11 \  
104.159996 1.131707 1.387571 1.322510 1.350930 1.223406  
104.169997 1.131587 1.387549 1.322884 1.350486 1.223658  
104.179998 1.131414 1.387530 1.323230 1.350018 1.223874  
104.189999 1.131319 1.387509 1.323601 1.349608 1.224129  
104.200000 1.131228 1.387509 1.323915 1.349214 1.224341  
  
 ... HKB:12-HKB:15 HKB:12-HKB:16 HKB:13-HKB:14 \  
104.159996 ... 1.071233 0.990922 0.411064  
104.169997 ... 1.071489 0.990873 0.410944  
104.179998 ... 1.071624 0.990832 0.410933  
104.189999 ... 1.071955 0.990734 0.410871  
104.200000 ... 1.072294 0.990736 0.410651  
  
 HKB:13-HKB:15 HKB:13-HKB:16 HKB:14-HKB:15 HKB:14-HKB:16 \  
104.159996 0.753501 0.910143 0.364396 0.539795  
104.169997 0.753502 0.909901 0.364396 0.539682  
104.179998 0.753522 0.909759 0.364316 0.539533  
104.189999 0.753529 0.909520 0.364281 0.539368  
104.200000 0.753482 0.909291 0.364271 0.539394  
  
 HKB:15-HKB:16 BS:0-BS:74 BS:0-BS:157  
104.159996 0.445009 1.046829 0.119864  
104.169997 0.445027 1.046903 0.119868  
104.179998 0.445038 1.046936 0.119734  
104.189999 0.445063 1.047000 0.119982  
104.200000 0.445110 1.046967 0.119830  
  
[5 rows x 122 columns]

### Involved devices (*S.dev*)

The *S.dev* command allows to obtain the complete list of devices involved in the serie and:

* the Name of the device used in the radio dataframe
* the Real device Id used during the measurement campaign
* The corresponding device Id used on the Body wear description
* At wich Subject the device is related.

Infrastrucure access point obviously don 't have related Subject.

>>> S.dev  
Name in Dataframe | Real Id | Body Id | Subject  
========================================================  
AP4 | 4 | HKB:4 |  
AP1 | 1 | HKB:1 |  
AP2 | 2 | HKB:2 |  
AP3 | 3 | HKB:3 |  
--------------------------------------------------------  
AnkleRight | 16 | HKB:16 | Nicolas  
KneeLeft | 14 | HKB:14 | Nicolas  
AnkleLeft | 15 | HKB:15 | Nicolas  
WristRight | 12 | HKB:12 | Nicolas  
WristLeft | 13 | HKB:13 | Nicolas  
ElbowLeft | 10 | HKB:10 | Nicolas  
HipRight | 11 | HKB:11 | Nicolas  
HeadRight | 5 | HKB:5 | Nicolas  
TorsoTopRight | 6 | HKB:6 | Nicolas  
TorsoTopLeft | 7 | HKB:7 | Nicolas  
BackCenter | 8 | HKB:8 | Nicolas  
ElbowRight | 9 | HKB:9 | Nicolas  
 | | |  
WristRight | 157 | BS:157 | Nicolas  
AnkleRight | 74 | BS:74 | Nicolas  
HandRight | 0 | BS:0 | Nicolas  
--------------------------------------------------------

## Accessing the data

In order to help people not familiar with the Pandas query format, some useful methods are provided in order to extract values from radio and non radio dataframes.

Get device position (*S.getdevp*)

The value of the device position at a specific time or range or time can be obtained by specifying:

* The device (Name in dataframe OR real id OR body id)
* The radio *techno* (Precising the techno is optional except when an ambiguity occurs, therefore error is raised)
* a given time in second or a [start time,stop time]. If no time is given, the position for all time stamps are provided

Hence, It is possible to get the positions of the HKB radio node 11 (Hip Right), between 5.0 seconds and 5.2 seconds with:

>>> Positions = S.getdevp(11,t=[5,5.2])  
>>> Positions  
 x y z  
5.000480 -0.139566 0.224905 1.016796  
5.010481 -0.139553 0.224845 1.016826  
5.020482 -0.139545 0.224825 1.016818  
5.030483 -0.139564 0.224730 1.016849  
5.040484 -0.139609 0.224642 1.016859  
5.050485 -0.139580 0.224613 1.016898  
5.060486 -0.139554 0.224586 1.016920  
5.070487 -0.139604 0.224492 1.016937  
5.080488 -0.139545 0.224452 1.016989  
5.090489 -0.139521 0.224391 1.016992  
5.100489 -0.139386 0.224397 1.016997  
5.110490 -0.139296 0.224315 1.017041  
5.120491 -0.139164 0.224189 1.017098  
5.130492 -0.138988 0.224128 1.017131  
5.140493 -0.138810 0.224048 1.017142  
5.150494 -0.138605 0.223969 1.017148  
5.160495 -0.138406 0.223877 1.017164  
5.170496 -0.138043 0.223803 1.017230  
5.180497 -0.137791 0.223654 1.017305  
5.190498 -0.137388 0.223580 1.017321

**NOTE : You may also obtain a classical numpy array instead of this Pandas object by using the "*values*" method :**

>>> Positions.values  
array([[-0.13956557, 0.22490462, 1.01679608],  
 [-0.13955284, 0.22484492, 1.01682581],  
 [-0.13954524, 0.22482529, 1.01681787],  
 [-0.1395645 , 0.2247298 , 1.01684918],  
 [-0.13960907, 0.224642 , 1.01685901],  
 [-0.13957962, 0.2246127 , 1.01689801],  
 [-0.13955351, 0.22458575, 1.01691986],  
 [-0.13960399, 0.22449205, 1.01693719],  
 [-0.13954485, 0.22445244, 1.01698865],  
 [-0.13952087, 0.22439058, 1.0169917 ],  
 [-0.13938625, 0.22439655, 1.0169975 ],  
 [-0.13929645, 0.22431535, 1.01704102],  
 [-0.13916449, 0.22418907, 1.0170979 ],  
 [-0.1389884 , 0.22412761, 1.01713135],  
 [-0.13880983, 0.22404759, 1.0171424 ],  
 [-0.13860497, 0.22396939, 1.01714777],  
 [-0.1384055 , 0.22387668, 1.01716443],  
 [-0.13804305, 0.22380293, 1.01722955],  
 [-0.13779123, 0.2236543 , 1.01730511],  
 [-0.13738791, 0.22358025, 1.01732141]])

### Get link value (*S.getlink*)

The value of a link *a* and *b* at a specific time or range or time can be obtained by specifying:

* The device (Name in dataframe OR real id OR body id)
* The device (Name in dataframe OR real id OR body id)
* The radio *technoa* and *technob* (Precising the techno is optional except when an ambiguity occurs, therefore error is raised)
* a given time in second or a [start time,stop time]. If no time is given, the position for all time stamps are provided

Hence, It is possible to get the HKB values between radio node 11 (Hip Right) and node 16 (Ankle Right) , between 5 seconds and 5.2 seconds with:

>>> Values = S.getlink(11,16,t=[5,5.2])  
>>> Values  
5.000500 -67  
5.010501 -67  
5.020502 -67  
5.030503 -67  
5.040504 -67  
5.050505 -67  
5.060506 -67  
5.070507 -67  
5.080508 -67  
5.090509 -67  
5.100510 -67  
5.110511 -67  
5.120512 -67  
5.130513 -67  
5.140514 -67  
5.150515 -67  
5.160516 -67  
5.170517 -67  
5.180518 -67  
5.190519 -66  
Name: HipRight-AnkleRight, dtype: float64

### Get link distance (*S.getlinkd*)

The ground truth distance separating a device *a* and device *b* at a specific time or range or time can be obtained by specifying:

* The device (Name in dataframe OR real id OR body id)
* The device (Name in dataframe OR real id OR body id)
* The radio *technoa* and *technob* (Precising the techno is optional except when an ambiguity occurs, therefore error is raised)
* a given time in second or a [start time,stop time]. If no time is given, the position for all time stamps are provided

Hence, It is possible to get the HKB values between radio node 11 (Hip Right) and node 16 (Ankle Right) , between 5 seconds and 5.2 seconds with:

>>> Distances = S.getlinkd(11,16,t=[5,5.2])  
>>> Distances  
5.000480 0.845013  
5.010481 0.845034  
5.020482 0.845045  
5.030483 0.845068  
5.040484 0.845090  
5.050485 0.845180  
5.060486 0.845229  
5.070487 0.845235  
5.080488 0.845309  
5.090489 0.845339  
5.100489 0.845353  
5.110490 0.845423  
5.120491 0.845482  
5.130492 0.845559  
5.140493 0.845563  
5.150494 0.845595  
5.160495 0.845602  
5.170496 0.845677  
5.180497 0.845769  
5.190498 0.845785  
Name: HKB:11-HKB:16, dtype: float64

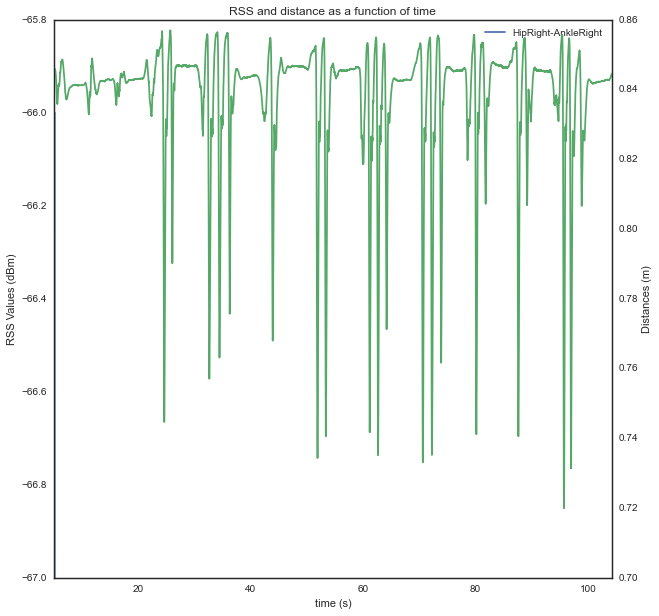
## Visualizing the Data

### Native Pandas Vizualization

Because radio data in CorSer are stored into Pandas objects, convenient vizualization method are directly available. Most of them can be found here : http://pandas.pydata.org/pandas-docs/stable/visualization.html

As an example, it is possbile tthe previous obtained values and distance with :

>>> # Ploting  
... ax=Values.plot() #plot values  
>>> l=Distances.plot(secondary\_y=True,ax=ax) # plot distances on the right side  
>>>   
>>> ##Labelling  
... ax.legend() # add legend box  
>>> ax.set\_ylabel( 'RSS Values (dBm) ') #set left ylabel  
>>> ax.right\_ax.set\_ylabel( 'Distances (m) ') #set right ylabel  
>>> ax.set\_xlabel( 'time (s) ') # set xlabel  
>>> ax.set\_title( 'RSS and distance as a function of time ')



In addition, CorSer also provides specific plotting methods which includes extra features.

### Plot method (S.plot)

The plot function allows to display the radio values of a link. The main parameters are always the same:

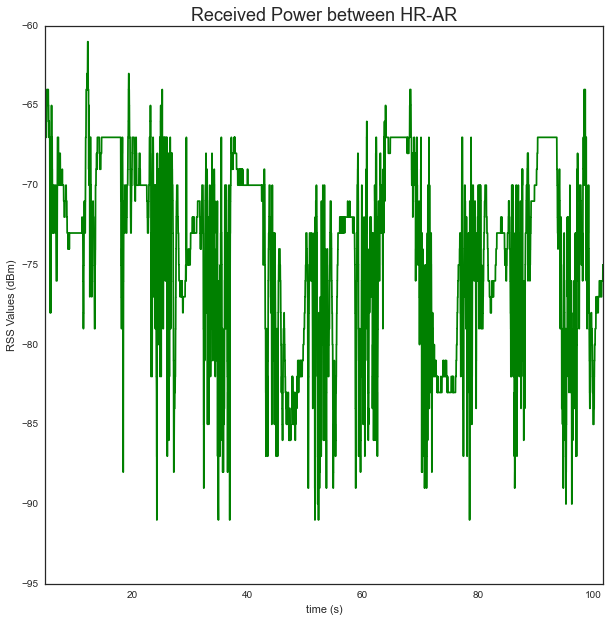
* The device (Name in dataframe OR real id OR body id)
* The device (Name in dataframe OR real id OR body id)
* The radio *techno* (Precising the techno is optional except when an ambiguity occurs, therefore error is raised)
* A given time in second or a [start time,stop time]. If no time is given, the position for all time stamps are provided

More option are availble, please refer to the docstring (*S.plot?*) for more information

#### Plot values

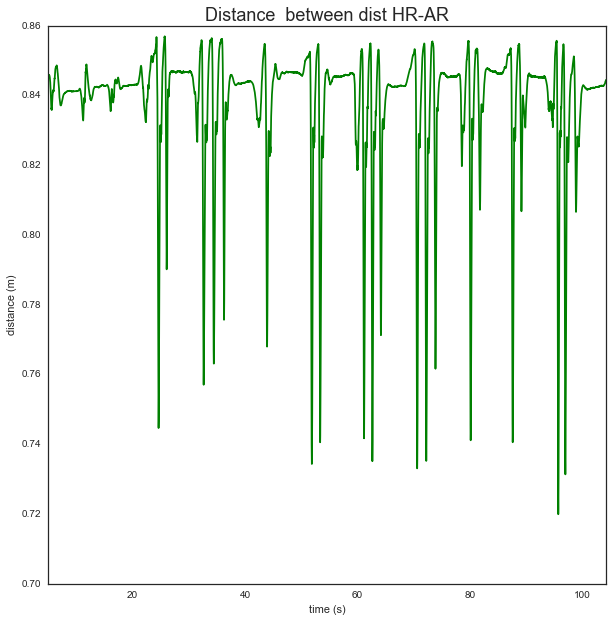
Continuying with the same example, it is possible to plot the HKB values between radio node 11 (Hip Right) and node 16 (Ankle Right) , between 5 seconds and 5.2 seconds with:

>>> S.plot(11,16,t=[5,5.2])

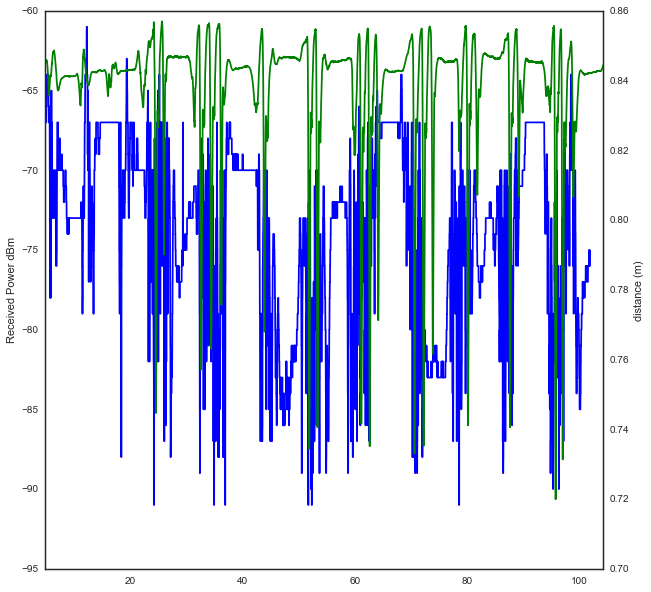


### Ploting evolution of the link distance

As well, it is possible to plot the distance using the *distance* parameter

>>> S.plot(11,16,t=[5,5.2],distance = True)  


It is also possible to get the same result than with the Pandas procedure with the following code :

>>> #plot value  
... f,ax = S.plot(11,16,t=[5,105.2],color = 'b ',title=False)  
>>>   
>>> # create right axis  
... ax2=ax.twinx()  
>>>   
>>> # plot distance  
... S.plot(11,16,t=[5,105.2],color = 'g ',title=False,  
... distance=True,  
... fig=f,ax=ax2)  


### Plot visibility method (S.pltvisi)

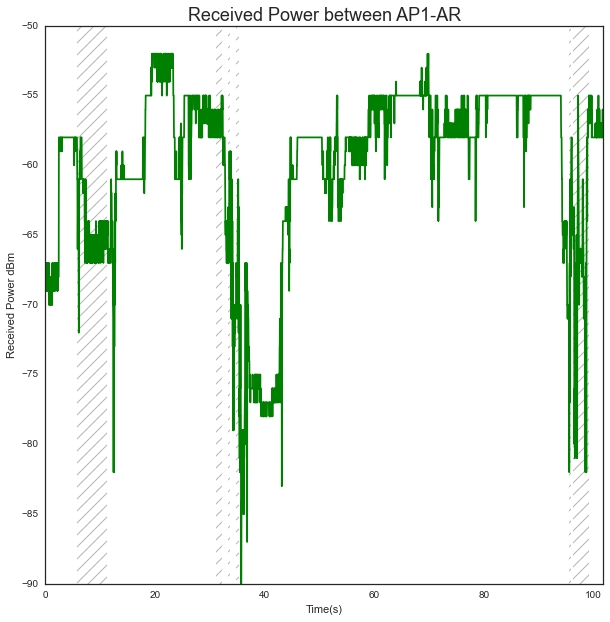
In order to go further in the radio value interpretation, it is convenient to have some extra information about the **optical visibility/occultation** of devices involved in a link.

This information allows to determine the line of sight (LOS) or non line of sight (NLOS) cases which are crutial for power level and delay interpretation.

This information can be superimposed to the radio values. To this end, the plot visibility (*S.pltvisi*) method is used. The **hatched** area denoted **NLOS** wheras **clear** area denotes **LOS**.

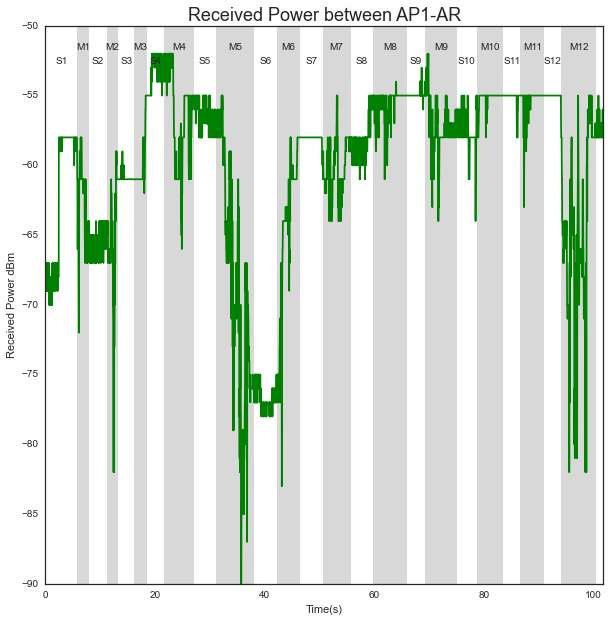
Parameters are the same than those the *plot* method:

>>> f,ax = S.plot(1,16)  
>>> S.pltvisi(1,16,fig=f,ax=ax)

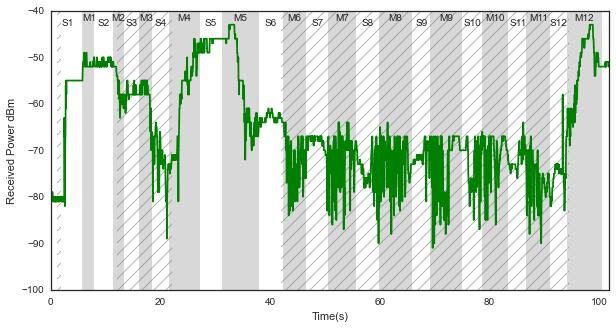


### Plot mobility method (S.pltmob)

As well it is possible to determine and indicate whether the subject is static or not by using the plot mobility method (*S.pltmob*). The succession of Static and Mobile sequences are denoted and resplectively, where is an index of the sequence.

>>> f,ax = S.plot(1,16)  
>>> S.pltmob(fig=f,ax=ax)  


The 2 above mentionned methods can also be used simultaneously as shown in the following example :

>>> # plot data in green)  
... f,ax=S.plthkb(1,13,figsize=(10,5))  
>>> # plot optical occultation (hatched lines)  
... S.pltvisi(1,13,fig=f,ax=ax)  
>>> # plot subject mobility (grey areas)  
... S.pltmob(showvel=False,ylim=([-100,-40]),fig=f,ax=ax)  


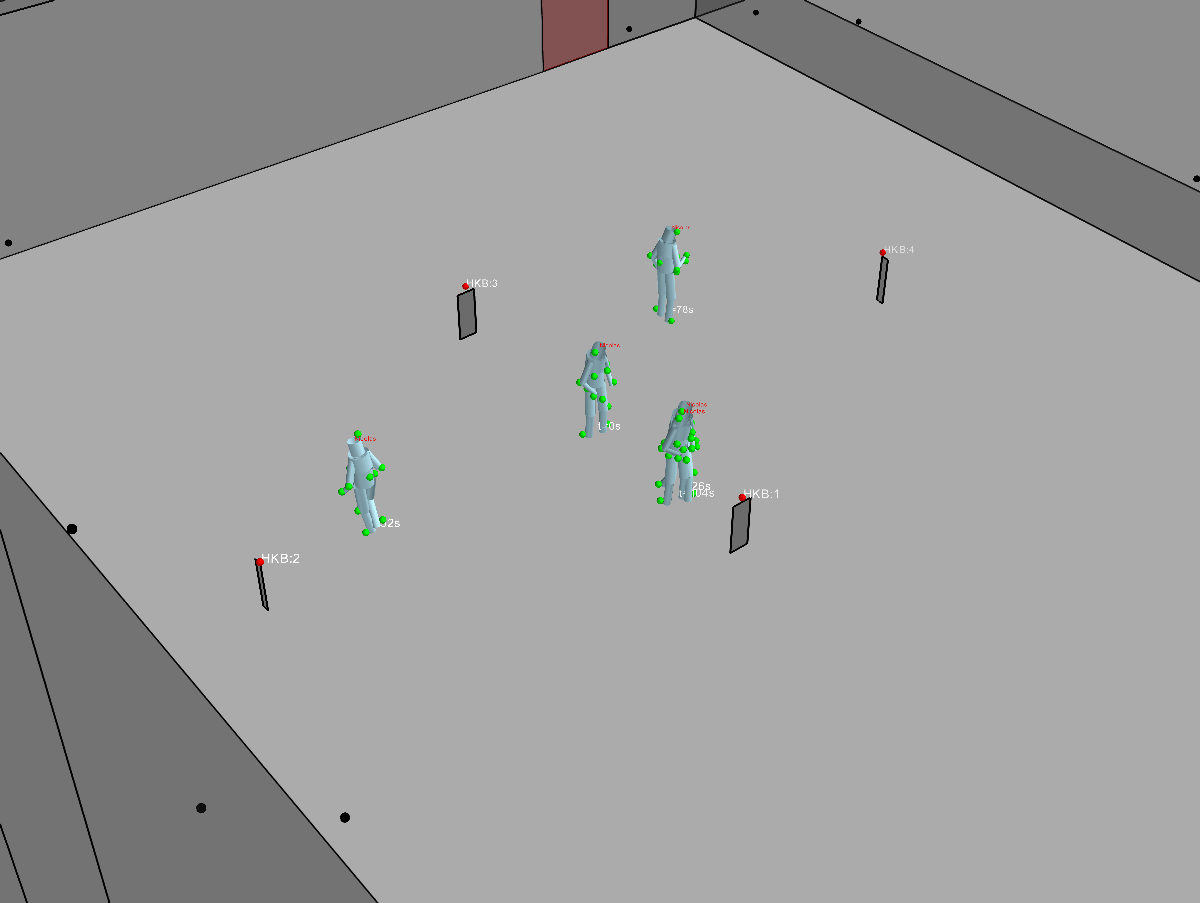
### 3D plot (S.\_show3)

With the help of the Mayavi Library, the CorSer class allows to display in 3D :

* The building where measurements have taken place
* The positions of Vicon Cameras
* The Multi-cylindric representation of the the subjects involved in the selected serie
* The position/ antenna pattern of the devices on the body(ies) and in the infrastructure.

By default, the use of the \*S.\_show3\* method display the complete scene with body(ies) and associated devices at 4 different timestamp

>>> S.\_show3()  
>>>   
>>> #the following line is only used to display in the notebook a screenshot of the mayavi window

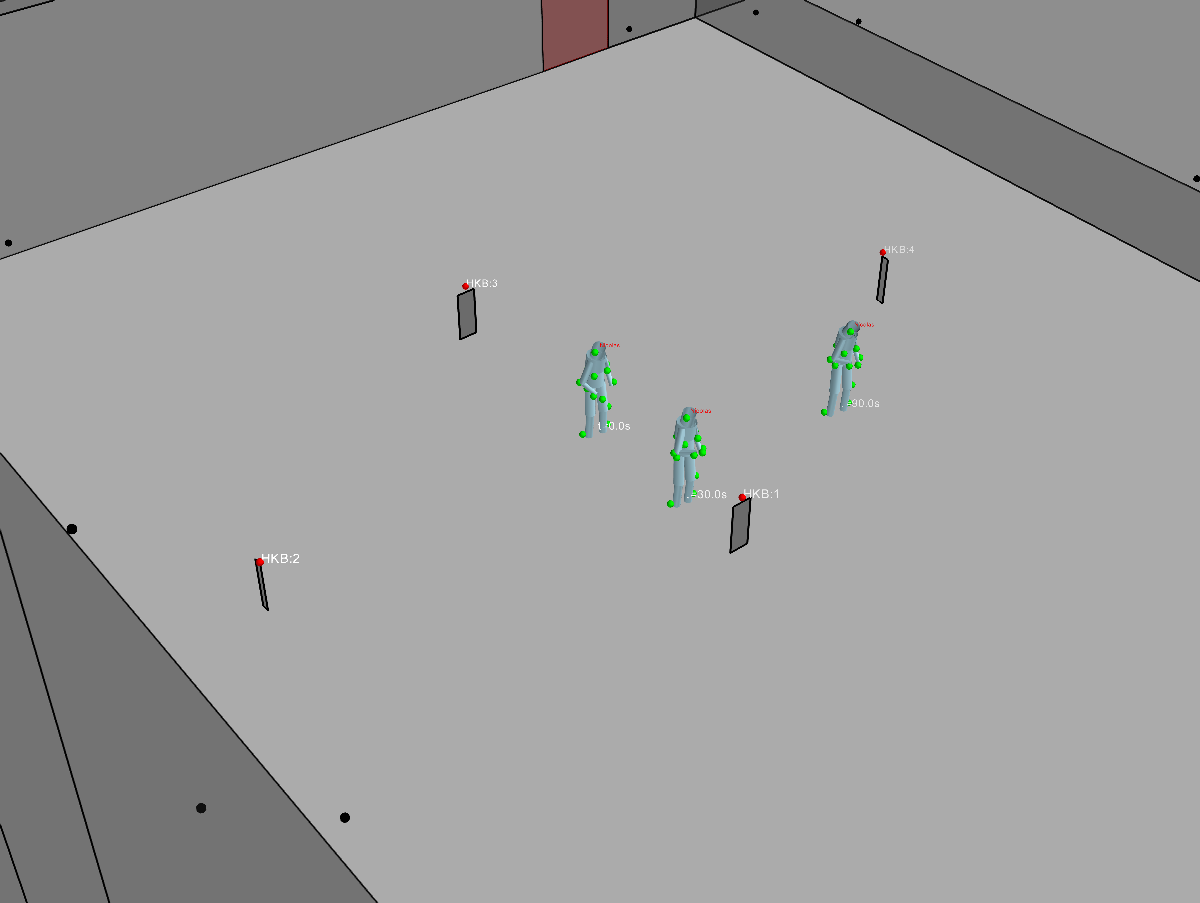


#### Specify time (*bodytime* parameter)

In order to display scene at specific timestamps, the parameter *bodytime* can be used

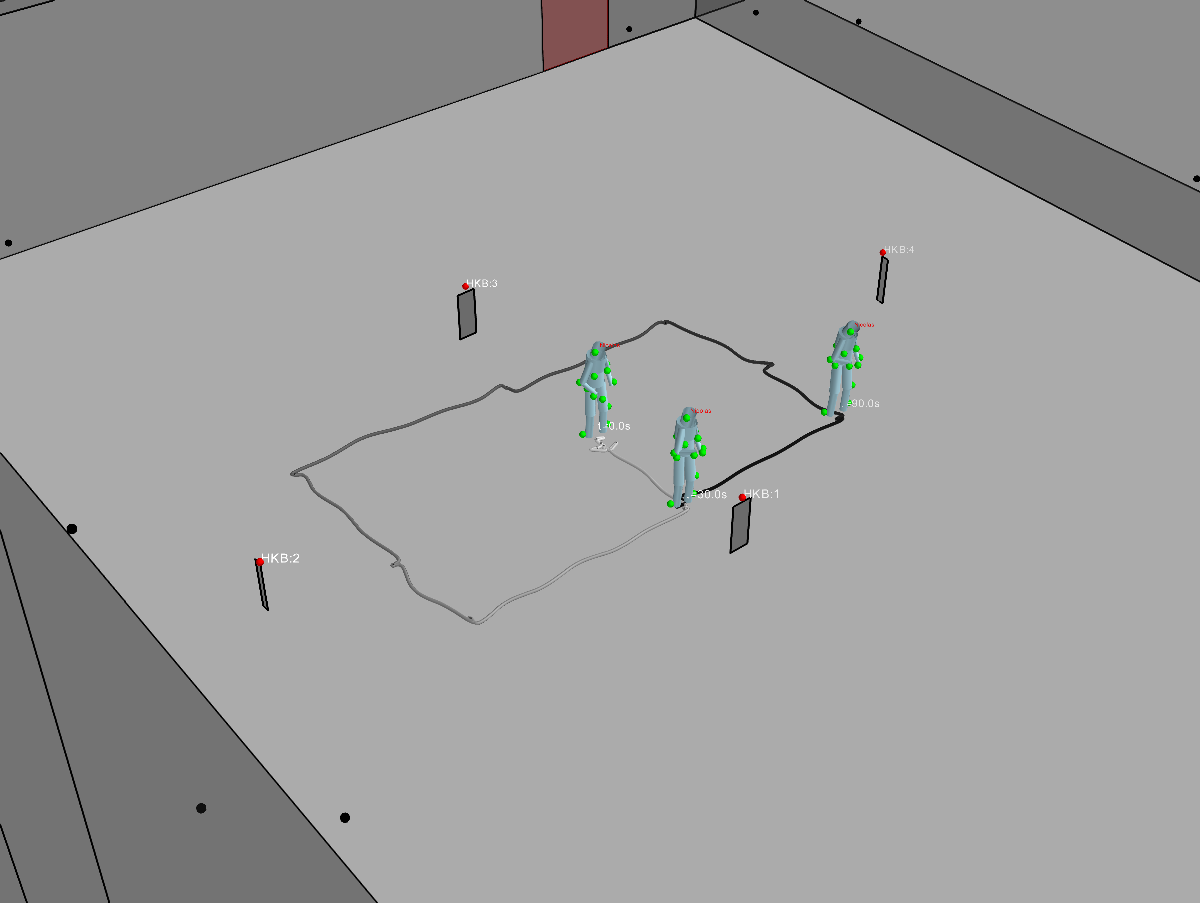
Example: to show the body position at , and .

>>> S.\_show3(bodytime=[0.,30.,90.])  
>>>



#### display trajectory (*trajectory* parameter)

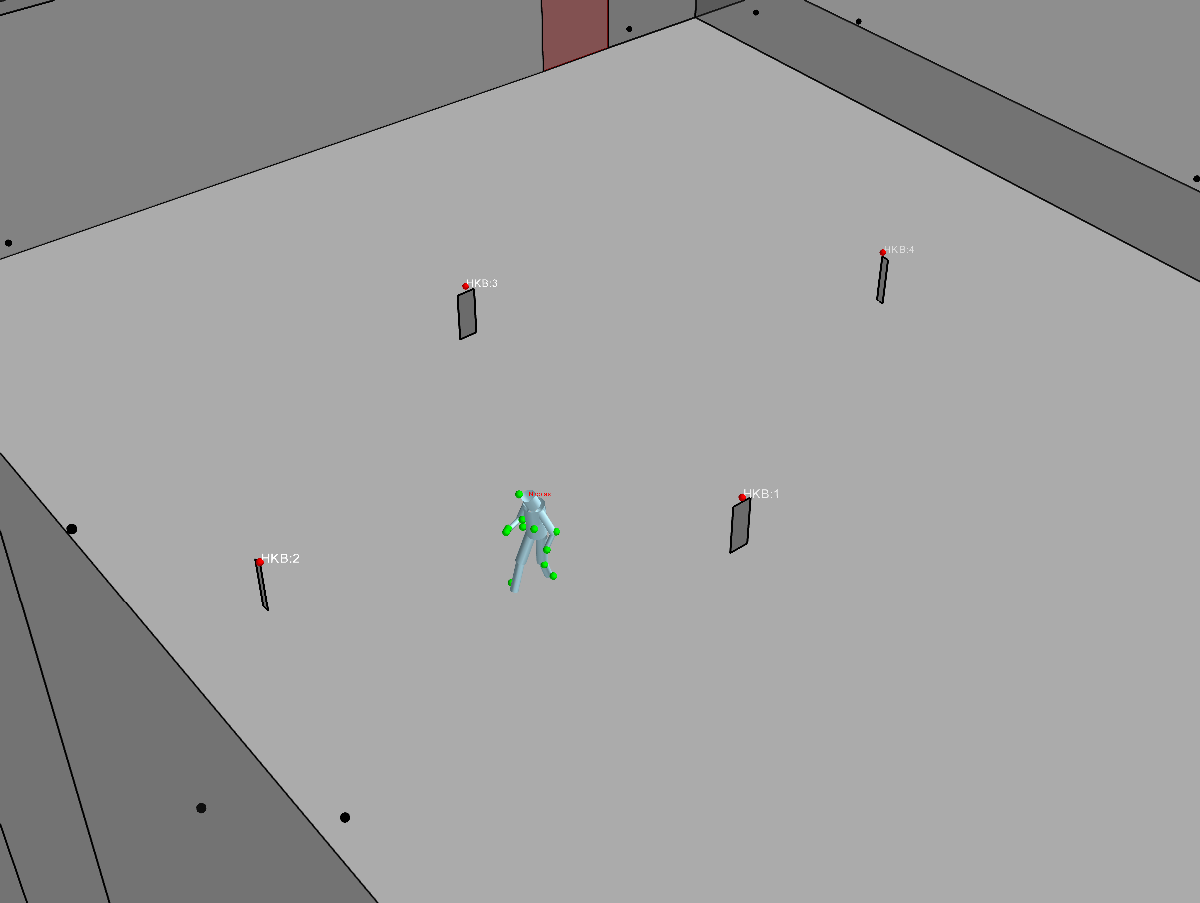
>>> S.\_show3(trajectory = True,bodytime=[0.,30.,90.])  
>>>



### 3D plot interactive (\*S.\_show3i\*)

The method \*S.\_show3i()\* allows to display the 3D scene with an extra window including a slider acting like a jog shuttle, to choose the timestamp to vizualize.

>>> S.\_show3i(t=35) #t=35 is an initialization value

### Interactive visibility (*S.imshowvisibility\_i*)

The visibility matrix can be displayed simultaneously to the 3D view.

For that purpose a visibility/occultation matrix is computed the first time the vizualization is called. The following code displays the tisibility matrix and associated 3D scene at the inital time

>>> S.imshowvisibility\_i(t=35)

