
test_coordonees_perceptives

Laurent Perrinet (INT, UMR7289)

March 12, 2014

```
In [18]: import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
In [19]: from parametres import VPs, volume, p, d_x, d_y, d_z, calibration, DEBUG
from modele_dynamique import arcdistance, orientation, xyz2azel, rae2xyz
```

```
In [20]: print xyz2azel.__doc__

renvoie le vecteur de coordonnées perceptuelles en fonction des
coordonnées physiques
```

```
xyz = 3 x N x ...
```

```
Le vecteur OV désigne le centre des coordonnées sphériques,
- 0 est la référence des coordonnées cartésiennes et
- V les coordonnées cartésiennes du centre (typiquement du
videoprojecteur).
```

```
cf. https://en.wikipedia.org/wiki/Spherical\_coordinates
```

```
i_vp = 1
```

```
In []:
```

1 Testing spherical coordinates

Plotting function:

```
In [21]: def plot_xyz2azel(motion, vp=VPs[1], axes_perc=['t', 'r', 'az', 'el', 'rec'], axes_xy
"""
Let's define a function that displays for a particular motion
(a collection of poistions in the x, y, z space --- usually
continuous) the resulting spherical coordinates (wrt to vp).

"""
fig = plt.figure(figsize=(18,10))
t = np.linspace(0, 1, motion.shape[1])[:, np.newaxis]*np.ones((1, motion.shape[2]))
rae_VC = xyz2azel(motion, np.array([vp['x'], vp['y'], vp['z']]))
motion_rec = rae2xyz(rae_VC, np.array([vp['x'], vp['y'], vp['z']]))
#print rae_VC.shape
for i_ax, axe_perc in enumerate(axes_perc):
    for j_ax, axe_xyz in enumerate(axes_xyz):
        #print i_ax, j_ax, 3*i_ax + j_ax
        ax = fig.add_subplot(len(axes_perc), len(axes_xyz), 1 + len(axes_xyz)*i_ax)
        if axe_perc == 't':
```

```

        #ax.plot(t, rae_VC[i_ax, :, :])
        #print motion_x[j_ax, :, :].shape, t.shape
        ax.plot(motion[j_ax, :, :], t)
    elif axe_perc == 'rec':
        ax.plot(motion_rec[j_ax, :, :], t)
    else:
        if axe_perc in ['az', 'el']: scale = 180./np.pi
        else: scale = 1.
        #print motion_x[j_ax, :, :].shape, rae_VC[i_ax-1, :, :].shape
        ax.plot(motion[j_ax, :, :], rae_VC[i_ax-1, :, :]*scale)
        ax.plot([vp[axe_xyz]], [0.], 'D')
        ax.set_xlabel(axe_xyz)
        ax.set_ylabel(axe_perc)
plt.show()
return rae_VC
print plot_xyz2azel.__doc__

```

Let's define a function that displays for a particular motion (a collection of positions in the x, y, z space --- usually continuous) the resulting spherical coordinates (wrt to vp).

```

vp = VPs[1]
In [22]: print vp
{'cz': 1.36, 'cy': 5.425, 'cx': 10.38, 'pc_min': 0.001, 'address':
'10.42.0.56', 'y': 5.19, 'x': 0.55, 'pc_max': 1000000.0, 'z': 1.36,
'foc': 21.802535306060136}

```

Pointing ahead:

```

print xyz2azel(np.array([vp['cx'], vp['cy'], vp['cz']]), np.array([vp['x'], vp['y'], v
In [23]: [ 9.8328086  0.02390186  0.          ]

```

Pointing left (bigger y) should give positive azimuth, zero elevation:

```

print xyz2azel(np.array([vp['cx'], vp['cy']+1, vp['cz']]), np.array([vp['x'], vp['y'], v
In [24]: [ 9.90727637  0.12498097  0.          ]

```

Pointing up (bigger z) should give positive elevation, zero azimuth:

```

print xyz2azel(np.array([vp['cx'], vp['cy'], vp['cz']+1]), np.array([vp['x'], vp['y'], v
In [25]: [ 9.88352796  0.02390186  0.10135186]

```

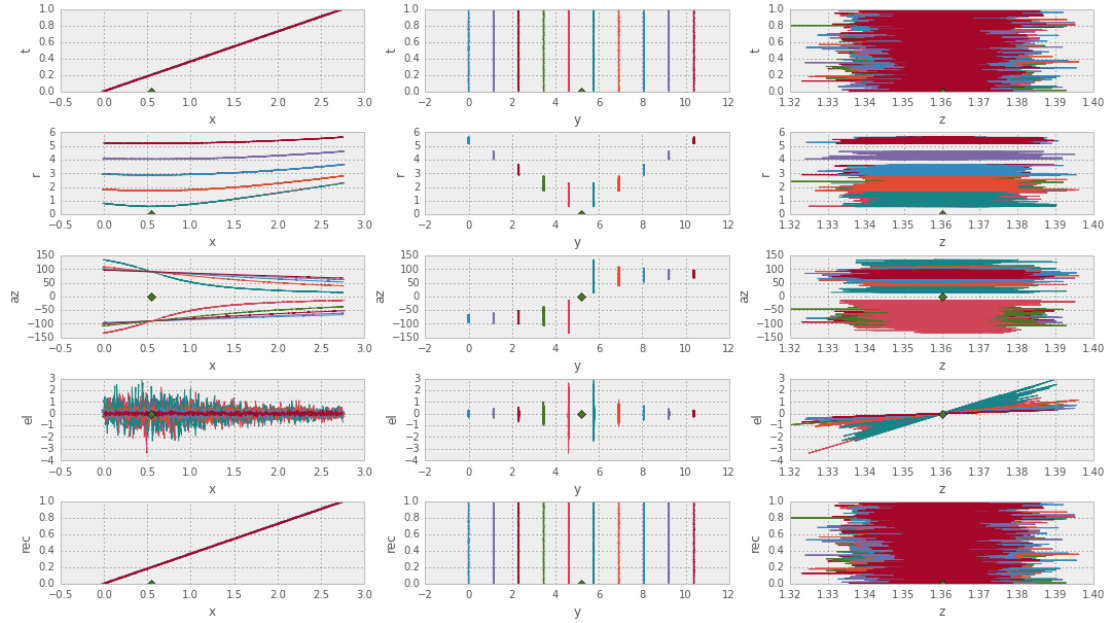
1.1 Motion in depth

Let's define N_player trajectories of N_t points, where players are distributed in the width (y) and move in the axis of the VP (x):

```

N_player, N_t = 10, 1000
In [26]: x = vp['x']*(np.linspace(0., 5., N_t)[np.newaxis, :, np.newaxis]*np.ones((1, 1, N_play
In [27]: y = vp['y']*np.ones((1, N_t, 1))*np.linspace(0, 2, N_player, endpoint=True)[np.newaxis
z = vp['z']*np.ones((1, N_t, N_player)) # constant
#print x.shape, y.shape, z.shape
motion_x = np.vstack((x, y, z))
motion_x += .01*np.random.randn(3, N_t, N_player)
motion_x.shape
rae_VC = plot_xyz2azel(motion_x, vp=vp)

```

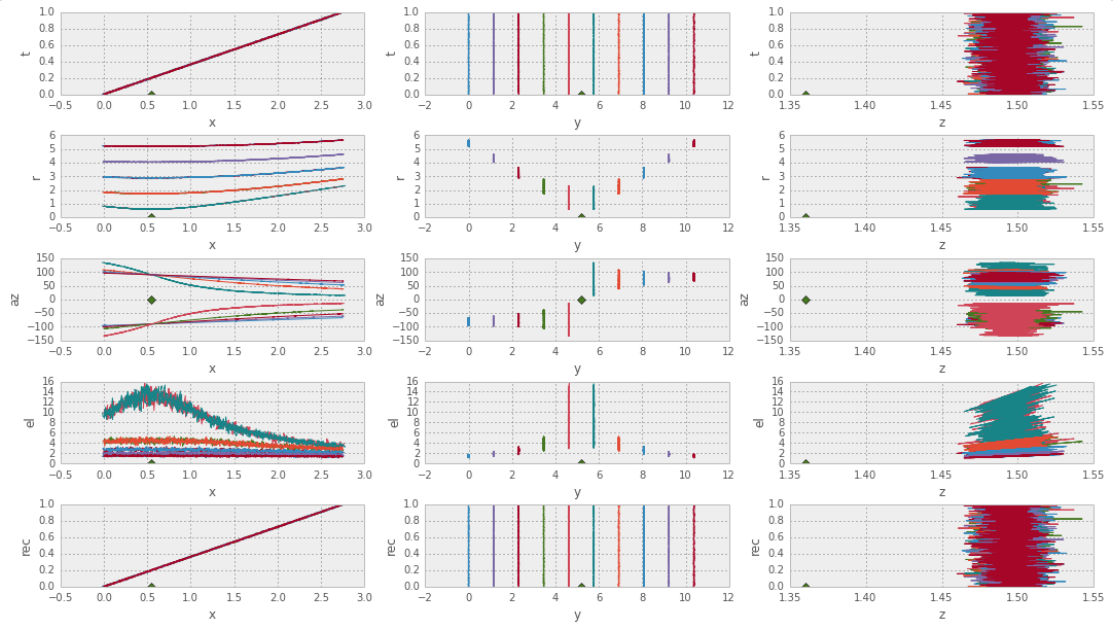


the same but with a motion slightly higher than the VPnote : when azimuth is superior to 90° in absolute value, it means it is behind

Note the small elevation:

```
In [28]: el = rae_VC[2,:,:]
print el.min(), el.max(), el.mean(), el.std()
-0.0590137586123 0.0511335003038 -1.04994607645e-05 0.00598766266327
```

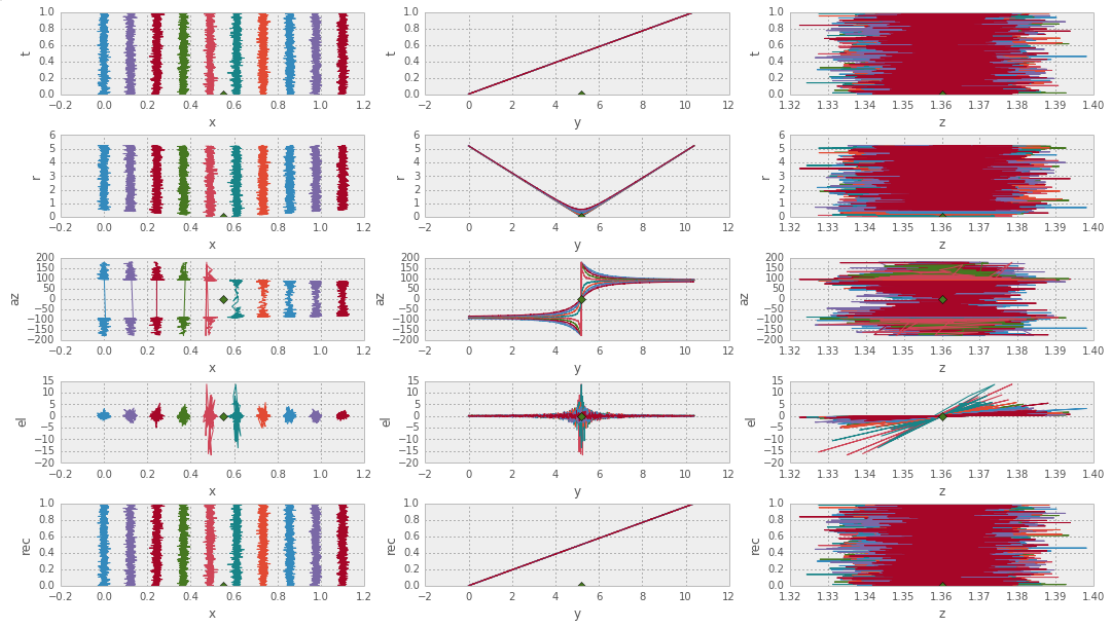
```
In [29]: z = 1.1*vp['z']*np.ones((1, N_t, N_player)) # constant
#print x.shape, y.shape, z.shape
motion_x = np.vstack((x, y, z))
motion_x += .01*np.random.randn(3, N_t, N_player)
rae_VC = plot_xyz2azel(motion_x, vp=vp)
```



being slightly up creates a small elevation, especially for a trajectory approaching the VP

1.2 Motion in width

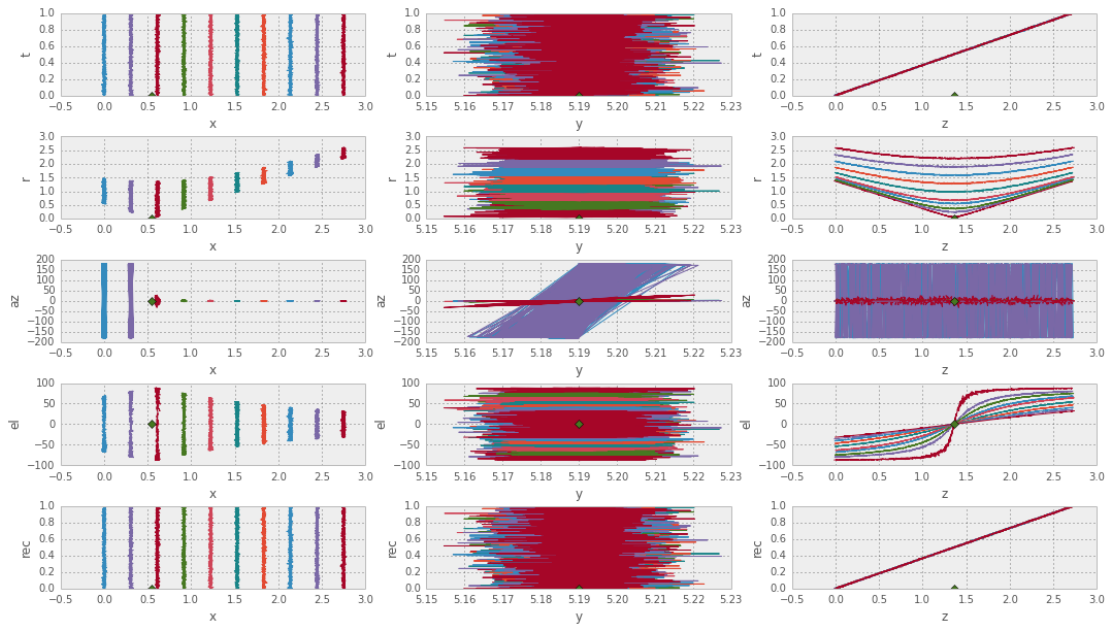
```
In [30]: x = vp['x']*np.ones((1, N_t, 1))*np.linspace(0., 2., N_player, endpoint=True)[np.newaxis]
y = vp['y']*(np.linspace(0, 2, N_t)[np.newaxis, :, np.newaxis]*np.ones((1, 1, N_player)))
z = vp['z']*np.ones((1, N_t, N_player)) # constant
motion_y = np.vstack((x, y, z))
motion_y += .01*np.random.randn(3, N_t, N_player)
rae_VC = plot_xyz2azel(motion_y, vp=vp)
```



note that when the motion is behind the observer ($x < VP[x]$), the azimuth flips from -180° to 180° , everything is normal...

1.3 Motion in height

```
In [31]: x = vp['x']*np.ones((1, N_t, 1))*np.linspace(0, 5., N_player, endpoint=True)[np.newaxis]
y = vp['y']*np.ones((1, N_t, N_player)) # constant
z = vp['z']*(np.linspace(0, 2., N_t)[np.newaxis, :, np.newaxis]*np.ones((1, 1, N_player)))
motion_z = np.vstack((x, y, z))
motion_z += .01*np.random.randn(3, N_t, N_player)
rae_VC = plot_xyz2azel(motion_z, vp=vp)
```



2 Testing great circle navigation

Similar tests, but now looking at the arcdistance and orientation functions:

```
In [32]: print arcdistance.__doc__
renvoie l'angle sur le grand cercle (en radians)

# rae1 ---> rae2

r = distance depuis le centre des coordonnées sphériques (mètres)
a = azimuth = déclinaison = longitude (radians)
e = elevation = ascension droite = latitude (radians)

http://en.wikipedia.org/wiki/Great-circle\_distance
http://en.wikipedia.org/wiki/Vincenty%27s\_formulae
```

```
In [33]: print orientation.__doc__
renvoie le cap suivant le grand cercle (en radians)

r = distance depuis le centre des coordonnées sphériques (mètres)
a = azimuth = déclinaison = longitude (radians)
e = elevation = ascension droite = latitude (radians)

http://en.wikipedia.org/wiki/Great-circle\_navigation
#http://en.wikipedia.org/wiki/Haversine\_formula
```

```
In [34]: def plot_xyz2perceptif(motion, vp=VPs[i_vp], axes_xyz=['x', 'y', 'z']):
    """
    Let's define a function that displays for a particular motion
    (a collection of positions in the x, y, z space --- usually
    continuous) the resulting orientation and arcdistance.
```

```

"""
fig = plt.figure(figsize=(18,10))
t = np.linspace(0, 1, motion.shape[1])[:, np.newaxis]*np.ones((1, motion.shape[2]))
rae_VC = xyz2azel(motion, np.array([vp['x'], vp['y'], vp['z']]))
motion_rec = rae2xyz(rae_VC, np.array([vp['x'], vp['y'], vp['z']]))
#print rae_VC.shape
for i_ax, axe_perc in enumerate(axes_perc):
    for j_ax, axe_xyz in enumerate(axes_xyz):
        #print i_ax, j_ax, 3*i_ax + j_ax
        ax = fig.add_subplot(len(axes_perc), len(axes_xyz), 1 + len(axes_xyz)*i_ax)
        if axe_perc == 't':
            #ax.plot(t, rae_VC[i_ax, :, :])
            #print motion_x[j_ax, :, :].shape, t.shape
            ax.plot(motion[j_ax, :, :], t)
        elif axe_perc == 'rec':
            ax.plot(motion_rec[j_ax, :, :], t)
        else:
            if axe_perc in ['az', 'el']: scale = 180./np.pi
            else: scale = 1.
            #print motion_x[j_ax, :, :].shape, rae_VC[i_ax-1, :, :].shape
            ax.plot(motion[j_ax, :, :], rae_VC[i_ax-1, :, :]*scale)
            ax.plot([vp[axe_xyz]], [0.], 'D')
            ax.set_xlabel(axe_xyz)
            ax.set_ylabel(axe_perc)
plt.show()
return rae_VC
print plot_xyz2azel.__doc__

```

NameError
call last)

Traceback (most recent

```

<ipython-input-34-6d7b7894cbc7> in <module>()
----> 1 def plot_xyz2perceptif(motion, vp=VPs[i_vp],
axes_xyz=['x', 'y', 'z']):
2     """
3     Let's define a function that displays for a particular
motion
4     (a collection of positions in the x, y, z space ---
usually
5     continuous) the resulting orientation and arcdistance.

```

NameError: name 'i_vp' is not defined

```
plot_xyz2perceptif(motion_x)
```

In []:

In []: