First steps with 5D coordinates and matlab

The EMA system initially uses a coordinate system that can roughly be defined as follows, assuming a subject is seated in the normal way in the EMA cube[1]:

Position (Cartesian):

x axis: anterior-posterior. Values increase to the front
y axis: left-right. Values increase to the left
z axis: up-down. Values increase upwards

Orientation (Spherical):

azimuth: Angle in the xy-plane, with zero along the positive x axis, and proceeding counter-clockwise. Like longitude on the globe. Range +/- 180deg.
elevation: Angle of elevation out of the xy plane. Like latitude on the globe. Range +/- 90deg

The origin (x=y=z=0) is at the centre of the cube.
Unless stated otherwise, positions will be given in mm, and orientations in degrees

Remember that there is one rotational degree of freedom that this kind of system cannot recover: When the sensor rotates about its main axis the induced signal does not change. The sensors are constructed such that the main sensor axis is perpendicular to the lead-wire.

Exercises

Exercise 1. Setting up a basic matrix of coordinates
In the matlab editor type the following line (without line-break):

```
sensornames=str2mat('t_back','t_mid','t_tip','ref','jaw','nose','upp
er_lip','lower_lip','head_left','head_right','mouth_left','mouth_rig
ht');
```

This creates a variable containing the names of the sensors for the 12 input channels of the hardware (as used in one of the Kleber/Reubold experiments).
Using
File > Save as
Store this in the m-file co_ex1.m
In the matlab command window, type

```
co_ex1
```

This should run the m-file and result in a string matrix being defined containing the names of the twelve sensors.
If you type

```
disp(sensornames)
```

---

[1] At the end of all the pre-processing we will actually use a different definition

you should see 12 lines.

We will now expand the m-file (i.e add the lines below in the fixed-width font) to set up a matrix with a rough estimate of the sensor positions and orientations for a typical experiment (this can be useful when we actually come to calculate the sensor positions).
Create a matrix with 12 rows (one for each sensor) and 5 columns (one for each coordinate):

```
data=zeros(12,5);
```

Initially all values in the matrix will be zero.
Assume the lower-lip has the following coordinates (x, y, z, azimuth, elevation):
[20 10 -10 90 0]
Insert this in the matrix:

```
data(8,:)=[20 10 -10 90 0];
```

Using the pdf file with photos of the sensor locations as guidance (camexp_S1_selphotos.pdf), work out the corresponding lines of code to define appropriate positions for the other sensors (the only sensor name that may not be self-explanatory is 'ref': this designates the sensor on the gums above the upper incisors).

Finally, include the code to store the two variables 'sensornames' and 'data' in a mat-file (named 'sensorpositions.mat'):

```
save sensorpositions sensornames data
```

Check this works properly by doing the following:
In the command window, run the m-file:

```
co_ex1
```

clear all variables:

```
clear variables
```

load the mat file:

```
load sensorpositions
```

Type `whos` to check the variables are present again.

The following exercises can be carried out either interactively at the command line or by adding to the m-file

Exercise 2. Useful matlab commands for displaying the data
(a) Text output in the command window of both variables

```
disp([sensornames num2str(data)])
```

(b) Plotting the sensor positions using typical matlab graphics functions

```
plot3(data(:,1),data(:,2),data(:,3),'.');
xlabel('x');ylabel('y');zlabel('z');
text(data(:,1),data(:,2),data(:,3),sensornames,'interpreter','none')
;
```

(use the figure's 'rotate' tool to get different views of the data)

Exercise 3. Adjusting an existing coordinate matrix
Assume it turns out that the x/y/z position of the lower lip is really at [30 20 10], and that all other sensors are shifted accordingly.
What's the best way of carrying out these shifts?

Exercise 4. Plotting the orientations: an alternative to spherical coordinates

Spherical coordinates can sometimes be awkward to use:

(i) It is awkward to calculate statistics etc. when almost the same angle can be referred to by very different values (e.g +179 and -179deg. are only 2deg. apart)

(ii) It is sometimes difficult to get an intuitive idea of what changes in the angles really mean in terms of change of orientation: For example, when the elevation is close to 90deg. huge changes in azimuth (e.g from 0 to 180deg) do not mean much change in where the sensor is pointing.[2]

So for some purposes it is convenient to convert the spherical coordinates to a set of 3 Cartesian coordinates. (Some of the displays used during the pre-processing show the information in this way. It can also be useful if you just forget how the spherical coordinates are defined.)

These coordinates can be thought of as defining the position in three-dimensional space of the point on the surface of a sphere (rather than using angles of longitude and latitude).

For our purposes we assume a sphere with a radius of 1.

matlab provides a ready-made function for doing this conversion: sph2cart.

As input it requires azimuth, elevation and radius (with angles in radians, so angles in degrees must be multiplied by pi/180).

So for the above example with the lower-lip (azimuth = 90, elevation = 0) we can write

```
[x,y,z]=sph2cart(90*pi/180, 0*pi/180, 1)
```

Convert the orientations of all sensors to the Cartesian representation.

Now plot the orientations using exactly the same procedures used above for the sensor positions.

If possible, work out how to just plot those sensors that have different orientation coordinates.

Exercise 5. Alternative orientations for tongue sensors

Sometimes the tongue sensors, in particular, may be attached to the tongue differently.

What would be the approximate orientation coordinates for the tongue sensors in the adjacent figure?

Work this out for both the spherical and Cartesian representation of the coordinates.

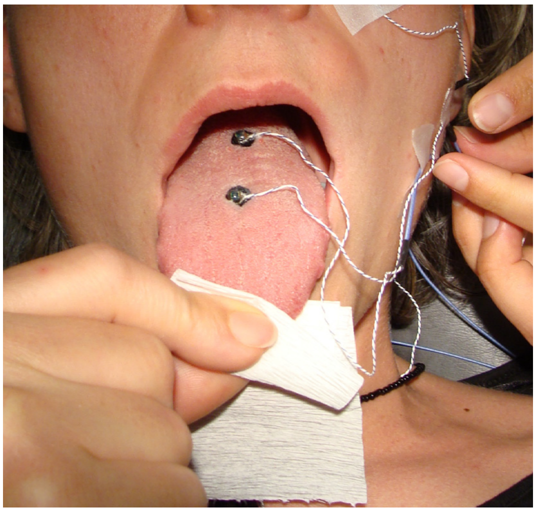
Choose elevation angles that fulfill the following conditions:

(1) For the more back sensor:

The sensor points up, looking from back to front

(2) For the more front sensor:

The sensor points down, looking from back to front



---

[2] The sph2cart function introduced in this section provides a way of checking this out for oneself; or just look at how lines of longitude converge near the poles of the globe.