Head movement correction

Introduction

In order to analyze speech movements it is necessary to factor out movement due to the head from the data of the sensors attached to the articulators. This task is usually combined with the definition of an anatomically-based coordinate system for the articulator movements. These tasks proceed in two main steps:

(1) Setting up of a reference object (wrapper script do_makerefobj.m)

(2) Re-mapping of all data with respect to this object (wrapper script do_rigidbodyana.m). This results in articulatory data expressed with reference to a (usually) skull-based coordinate system. At the same time the head movement data itself is stored as new separate datasets.

If the data for the definition of the reference object is to be based on a specially recorded trial in which, for example, the occlusal plane of the subject is captured, then the position data for the relevant sensors will first have to be determined as a preliminary stage. Since this is probably the most common situation we will go into this first. We will assume that it involved placing two sensors on a plastic triangle, with the recording made as the subject bites on the triangle, and with the two sensors aligned with the midline of the subject.
The alternative procedure is to base the reference object on data from one of the standard trials in the session, preferably one in which the subject is not moving and has the head in a neutral position (we often refer to these as 'rest' trials). After discussing the definition of a reference object for the standard case based on an occlusal plane trial, we will give some indications as to how the reference object can be defined if based on a 'rest' trial.

Position calculation for occlusal plane trials

Sometimes these trials are recorded in a separate subdirectory from the main data, often called something like 'palocc'.
If not, it a good idea to make a subdirectory called 'palocc' now, and copy the relevant amp files into a new subdirectory 'amps' below 'palocc'.
Then copy do_filteramps.m, do_tapad_ds.m and the mat-file of start positions used with do_tapad_ds into 'palocc'.
These files now need to be edited as follows (to avoid inadvertently overwriting other copies of the file it is worth renaming them with '_occ' as suffix, e.g do_filteramps_occ):

do_filteramps_occ: Edit the list of sensors so that names like 'occ1' and 'occ2' are given to the appropriate channels (this could have been done already in the original do_filteramps if the channels used for the occlusion sensors were not required for articulator movement in the main part of the experiment).
Similarly, edit the filter assignments. Put the occlusion sensors in the same list as the reference sensors (actually, this is not critical because the reference object procedure takes the average value of each sensor over the whole trial).
Set the dodown flag to 1
Edit the list of trials
Edit the list of channels (not crucial, but makes the "pegelstats" display easier to read)

do_tapad_ds_occ: Edit the list of channels (only the reference sensors and the occlusion sensors need to be processed), and the list of trials.
Also edit the name of the mat file containing the start positions (e.g 'startpos_v1_occ')

startpos_v1_occ.mat: This needs modifying by hand because although values present for the reference sensors should be basically OK for use here any values present in the occlusion sensor channels will almost certainly be inappropriate. The steps to be followed here are actually a good illustration of how the file of start positions can be modified manually for other exceptional cases (we assume the mat file has been set as explained in section ??). We assume for the example in this document that the occlusion sensors occ1 and occ2 are on channels 12 and 11, respectively, and that occ1 is the more posterior of the two. matlab commands are in the fixed-width font:

```
clear variables
load startpos_v1_occ
```

set these sensors to zero position and orientation

```
data([11 12],:)=0;
```

It should be possible get a rough idea of the azimuth angle from inspecting how the sensors were attached to the plastic triangle. A rough setting could be:

```
data(11,4)=-90;
data(12,4)=90;
```

Add an additional comment to the comment variable (without line break)

```
comment=['Modified with preliminary settings for
occlusion sensors' crlf comment];
```

For perfect documentation also

```
label=str2mat(label(1:10,:),'occ2','occ1');
```

And finally

```
save startpos_v1_occ
```

Run do_filteramps_occ and then do_tapad_ds_occ.
It can happen that the position calculation may not succeed for all channels (tapad's statistics show all NaNs). In that case it will be necessary to edit the start positions by hand again to give a more realistic value. If one of the occlusion sensors fails, for example, but the other is successful, then it should be possible to work out quite accurately what start positions for the failed one should be.
Once the positions have been calculated it is a good idea to have a quick look at them with do_do_comppos just to make sure that the data is reasonably stationary over the whole trial, and that it is not located in an unusual part of the measurement field (if several occlusion trials have been recorded this information could help when deciding which one to use).[1]

[1]For these kind of trials, which involve essentially stationary data, we are assuming that there is no necessity for carrying out parallel processing with tapad and kalman, nor for carrying out amplitude adjustment to help eliminate occasional outliers. Amplitude adjustment can certainly not be applied to the occlusion sensors because stationary data does not provide suitable material for the regression analysis (do_ampvsposamp). Whether the

Construction of reference objects

Coordinate systems

First it is necessary to decide what overall coordinate system is to be used.
Recall the arrangement of the x, y, z axes used up to now (assuming subject is seated in the usual position within the transmitter cube):

- X: Anterior-posterior. Positive X direction to the front, i.e in the direction the subject looks
- Y: Transversal (Lateral). Positive Y axis to the subject's left
- Z: Longitudinal (vertical). Positive Z direction upwards

However, there is no God's truth about this particular arrangement. Ultimately, the arrangement used is a matter of personal preference or habit.
We have always used a system in which the anterior-posterior axis increases to the ***rear***. This was the case with the Carstens 2D system AG100, and has the advantage of corresponding to the arrangement of places of articulation in the IPA chart.[2]
Re-mapping the coordinate system can be carried out as part of the definition of the reference object (in addition to anchoring the coordinate system with respect to skull-based criteria). Basically it involves a rotation by +90 degrees around the z-axis. This gives the following new definition of the x and y axes (z remains unchanged):

- X: Transversal (Lateral). Positive X axis to the subject's left
- Y: Anterior-posterior. Positive Y direction to the rear

Interactive creation of the reference object

This is carried out with the small interactive function makerefobjn.
This is best called from a small wrapper script named do_makerefobj.m.
This needs to specify the sensors to use, e.g

```
sensorlist=str2mat('occ2','occ1','ref','nose','head_left','head_righ
t');
```

and the file from which to take the position data, e.g

```
posfile=['ampsfiltds' pathchar 'beststartl' pathchar 'rawpos'
```

---

amplitude adjustment that has usually already been set up for the other sensors should be applied to them in the occlusion trials too is less clear (and has never really been checked). My guess is that it should make very little difference to the head movements ultimately factored out of the articulatory data. This would require detailed analysis of the head movement parameters and the taxonomic distance output by rigidbodyana (see below). If one does want to try amplitude adjustment for the occlusion trials, then it is absolutely essential to make sure that the two occlusion sensors are not included in the sensorlist input argument (in do_adjamps). Then their amplitudes will simply be copied unchanged from the input amp subdirectory to the new one where the output is stored.

[2]Another incidental reason for retaining this arrangement with the 3D system was that at the time of our first work with the system we were also analyzing 3D MRI data that followed this arrangment. Note that a lot of North American work, particularly that based on the Perkell MIT magnetometer system arranges the anterior-posterior axis to increase towards the front. In figures in publications it is not always obvious at first sight which arrangement is being used.

```
pathchar '0481'];
```
There is also an input argument ‘vdistance’ that should be set to 50. This is related to the concept of ‘virtual sensor’ that is explained below.
The interactive function is then called with
```
makerefobjn(posfile,sensorlist,vdistance)
```

After starting the program the first prompt to be seen is
```
>> COM file name (<CR> to ignore) :
```
Here a file containing commands for the program can be specified. This is useful to ensure that one applies exactly the same transformation to all subjects in an experiment (or to quickly process several different occlusion trials available for one subject). This can conveniently be based on the log file that will be described below.
The program is based on a small set of keyboard commands. These can be displayed by typing ‘h’ for ‘help’:
```
o: Choose sensor to locate at origin
O: Choose single sensor coordinate to locate at origin
r: Set second sensor at desired angle relative to first sensor in
desired plane
a: Rotate all sensors by desired angle in desired plane
x: Reset sensors to original position
k: Enter keyboard mode
l: List current coordinates
c: Change description of transformed coordinate system
e: Store results and exit
```

The next figure shows a typical display immediately after the start of the program, i.e. it shows the sensors in the original coordinate system, just as recorded.

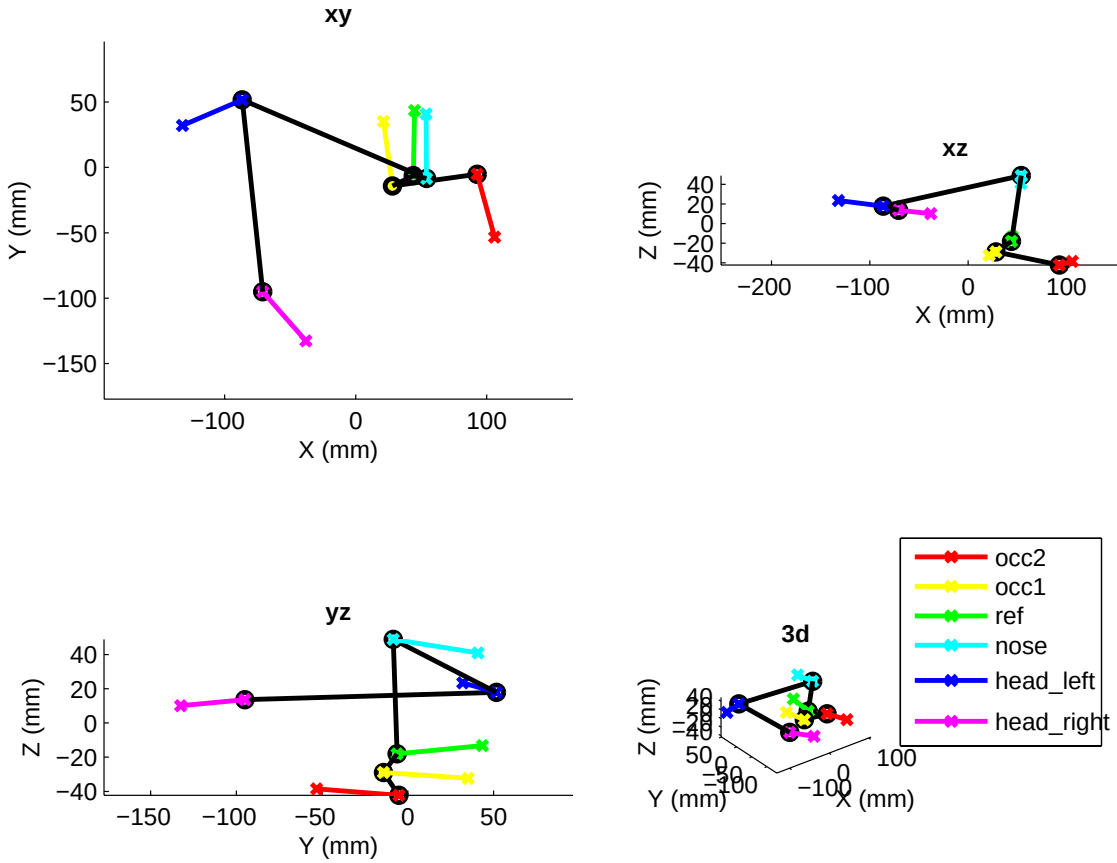

Figure 1 Reference sensors and occlusion sensors in the original coordinate system

The actual sensors are at the black circles, and joined by black lines. The order of the sensors in the input argument sensorlist determines the order in which the sensors are joined by black lines, so adjusting the order in the list can make the display easier to follow (that was the reason for the order given above).
The coloured lines introduce the concept of ***virtual sensors***. The position of the crosses at the end of the coloured lines is determined by the spherical coordinates of the corresponding sensor at the black circle at the other end of the line, together with the fixed distance of 50mm specified in input argument vdistance (i.e. all coloured lines are exactly this length). This provides a way of making use of the sensor orientation information within a purely Cartesian framework both in the current program and also in the next step (rigidbodyana). [3] These virtual sensors can be referred to like normal sensors in any of the following

[3]The reason for this 'sleight of hand' is that there are ready-made algorithms in the literature for calculating coordinate transforms of this kind based on 3D Cartesian coordinates. It should be possible to reformulate the problem to use the orientation angles directly. The choice of 50mm for the distance between 'parent' sensor and virtual sensor is somewhat arbitrary. The original motivation was that the noise in the 3D coordinates of the virtual sensor should be similar to that of a normal sensor, but this has not been followed up extensively.

commands by prefixing ‘v_’ to the sensor name. [4]

We will now walk through the typical series of steps to define a coordinate transform based on an occlusion trial (user input in the fixed-width font).

Define the origin to be at the more posterior occlusion sensor (usually located just inside the mouth):

```
o
occ1
```

Observe how all panels of the figure are updated after each command.
The sensor occ1 should provide a well-defined origin for the vertical (up-down) axis as well as for the lateral (left-right) axis, but less so for the anterior-posterior axis. This latter dimension depends more strongly on precisely how the sensor is attached to the plastic triangle, and also to how firmly it is pressed into the mouth. The sensor attached to the gums of the upper incisors usually provides a good origin for the anterior-posterior axis. This remains true even if it was not possible to glue it precisely on the mid-line. As the final step, we will redefine the origin of this axis (it is better to do it at the end, because all the rotations involve occ1, and if occ1 is not at the origin the rotations may shift occ1 away from the origin, even if it was placed there initially).

The main job now is to define the orientation in each of the three planes (xy, xz and yz), corresponding to the three main panels in the figure (the panel labelled 3D is usually not very useful). We will start with the xy plane as this is where the really big change occurs.
In this plane occ1 is probably at an angle of roughly 180deg relative to occ2. Specifying a rotation so that occ1 is at 90deg relative to occ2 will carry out the transformation to the new overall coordinate system and also hopefully align the anterior-posterior axis with the mid-line of the subject:

```
r
occ2
occ1
90
```

Check that the head sensors now show the most strongly positive values on the y axis. occ2 should be located at a negative value, and the other sensors (ref, nose, occ1) at around zero.
We will now carry out the rotation in the yz plane. The panel labelled ‘yz’ now corresponds to a sagittal view of the data. It is the rotation in this plane that actually defines the occlusal plane. Set occ1 at 0deg relative to occ2:

```
r
yz
occ2
occ1
0
```

Normally, this results in a change of only a few degrees.
The last rotation, in the xz plane, is probably the least critical in terms of the phonetic interpretability of the data. Note that the xz plane now shows the positions as if you are looking at the subject from the front (so subject left side is on the right in the xz panel).
To define this rotation we preferably need two sensors on the midline and widely separated

---

[4]Whenever the program prompts for the name of a sensor the list of available choices can be obtained by typing ‘?’.

on the z-axis. The most obvious choice is occ1 and nose:

```
r
xz
occ1
nose
90
```

Again, this will probably only give a change of a few degrees.
As the last step we re-define the origin for the anterior-posterior axis with the following commands (note capital O compared to lower-case o before):

```
O
ref
2
```

This completes the definition process. The following figure shows the resulting configuration.

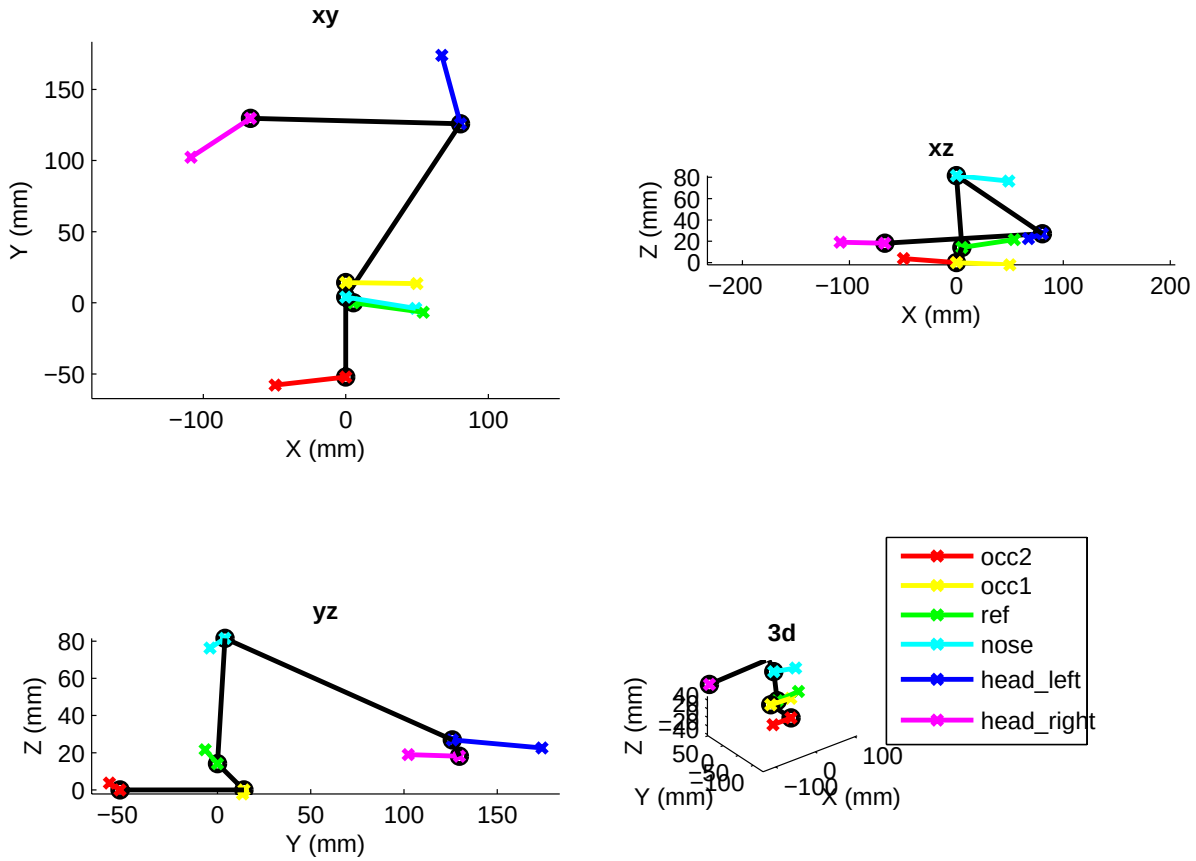

Figure 2 Reference sensors and occlusion sensors after transformation of the coordinate system

The results are stored by choosing command '`e`'. The current coordinates are displayed in the command window and the program waits with the keyboard prompt '`K>>`'. Type '`return`' to continue. The user will be prompted to write a comment to be stored with the reference object. If there were no special features (e.g any of the sensors normally used were missing or unreliable) then this can be left empty.

Notes on the output of makerefobjn
The main output file is a mat-file for use by the next procedure rigidbodyana.m. It is stored in the same location as the input position file with the suffix '_refobj' added, e.g ampsfiltds/beststartl/rawpos/0001_refobj.mat
The other output is a text file (log file) called makerefobjlog.txt. This contains a listing of all the commands used (with the user prompts included as comments preceeded by '%'), as well as a listing of the final sensor coordinates.

Detailed information on how the log file can be used
The log file can be used as the basis for a command file for running the procedure again non-interactively: Copy all the material up to just before the 'e' command into a separate text file named e.g 'makerefobj_commands.m' (it's better not to include the 'e' command in the command file so that a check can be made that nothing has gone wrong before storing).
It can also be useful to paste the listing of the sensor coordinates into another additional text file (e.g named 'refobj_coordinates.txt'), as this gives a convenient quick reference to the result of the definition procedure (and is also useful if several occlusion trials are available that one wants to compare).[5] This gives another chance to check that the definition process has worked out as intended. In particular, check that coordinate values of zero occur where expected. Further examples: (1) If you know that the ref sensor on the upper incisors was glued somewhat to the left of the midline (refer to photos of the session) then check that the x-coordinate is slightly positive. (2) We don't have any precise information on how well the midpoint between the head_left and head_right sensor generally corresponds to the midline of the tongue etc. But if the final x-coordinates of the head sensors are very asymmetrical this raises the possibility that the occlusion sensors may not have been aligned very well with the subject's midline.
Note that the log file itself is appended to (rather than overwritten) if multiple runs of the reference object procedure are carried out (in the same subdirectory).

Documenting non-standard coordinate transformations
The program automatically includes the following comment in the output mat-file to document the transformation from the original AG500 coordinate system to the one we have normally used for further work:

'Coordinate system resulting from transformation:'
'Skull-based'
'x: Transversal (Lateral). Increases from right to left'
'y: Anterior-Posterior. Increases from front to back'
'z: Longitudinal. Increases from low to high (foot to head)'

If you decide to define the transformation differently (or decide to stay with the basic AG500 system) then a comment to this effect should be included using the 'c' command.

Using articulator sensors rather than special occlusion sensors
Here we sketch out a possible procedure if the reference object is to be based on a 'rest' trial,

[5]The graphics can also be stored for documentation (not done automatically): saveas(gcf,'refobj.fig','fig')

using standard reference sensors (ref (upper incisors), nose, head_left, head_right), plus normal articulator sensors on tongue and lips.
We will assume the simplest case where the ref sensor is located on the midline (otherwise lip or jaw sensors may have to be used for some steps).

origin:	Set the origin to ref
xy plane:	Define this big rotation by setting tongue-back at 90deg relative to ref (alternative: set head_left at 0deg relative to head_right, if photos suggest a comparable location in the anterior-posterior dimension).
yz plane:	Set nose at 90deg relative to ref[6]
xz plane:	Set nose at 90deg relative to ref

Inadvertent shifts in the origin
Depending on what sensor defines the origin, and what sensors define the rotations the origin may get shifted during the rotations, so it is advisable as a final step to simply repeat the origin command.

Potential use of virtual sensors
The virtual sensors are always displayed, even if they are not actually required for the definition of the reference object, since they are invariably used when the sample-by-sample mapping of head position to the reference configuration is actually carried out by rigidbodyana. So it is always a good idea to check that the display of the virtual sensors corresponds to any information that can be derived from photos etc.
However, some situations are conceivable in which they could in fact be used in the definition of the reference object. For example, consider a situation in which the nose sensor is unavailable. This makes defining the rotation in the xz plane difficult. However, a workaround is to use, for example, the 'virtual' occ1 sensor (assuming the arrangement as shown in the figures), setting it a 0deg relative to the actual occ1 sensor:

```
r
xz
occ1
v_occ1
0
```

(This may not be very accurate because in practice it is not possible to attach the sensor to the plastic triangle with its main axis exactly parallel to the surface of the triangle. But instead of setting the angle to precisely zero, one can also aim for a compromise solution using occ2 and v_occ2 as well.)

[6]Sometimes it may be the case that one has occlusal plane data for a subject available from an earlier experiment. This earlier data could in principle be used to determine the angle between the occlusal plane and the line joining ref and nose (the function prints out the amount of any rotation carried out). One could then specify that angle for the present scenario, rather than using exactly 90deg.