Amplitude Adjustment

Basic concept: posamps
More background: Coming soon

The amplitude adjustment procedure is based on a regression analysis which aims to predict the residual for each sensor-transmitter pair from all transmitter signals for that sensor.
It is important to base this regression analysis on data that is likely to be reliable.
Accordingly, the first stage is to use do_do_comppos to collect various statistics on the data, so that these can then be used to eliminate unusual data from the regression analysis.

These procedures should ideally be done separately for the tapad and kalman data.
We will explain the procedures for the tapad data first

Set doshowtrial to 0
Set basepath to point to the tapad solution
Comment out altpath
Enter trials that should be excluded from the statistics calculations in restlist

> The regression analysis requires data that is representative for the experimental tasks that one actually wants to analyze. So put into this list any trials where the subject does anything unusual not related to the purpose of the experiment, e.g large body movements with no speech, and conversely, also those where the subject is neither moving nor speaking at all.

Choose a value for compsensor

> This should be the number of a sensor for which the position calculation appears to have given stable results. The Euclidean distance will be calculated between this sensor and all other sensors. Normally a reference sensor should be chosen (the reason for this will become apparent shortly). The first choice is generally the sensor on the bridge of the nose (it is generally reasonably in the centre of the measurement field, and unlike a sensor on the upper incisors it is not in the nasty warm, moist environment of the mouth). In unclear cases it will be necessary to proceed by a process of elimination to determine which reference sensor has overall the most stable distance to the other reference sensors.

Make sure that the specification for diaryfile is not commented out.
Set autoflag to 1 or 2 because here we are interested in the behaviour of each sensor over the whole session, rather than in specific trials.

do_do_comppos computes statistics on rms, tangential velocity and euclidean distance from the comparison sensor. These are displayed on the screen as the program is executed, but can also be retrieved afterwards from subdirectory ‘figs/’.
For example, the results for sensor 1 can be retrieved with

```
open figs/comppos_stats_ds_beststartl__comp6_trialstats_1.fig
```

(the precise name may vary somewhat depending on the location of the input data; this example assumes compsensor was set to 6)
In each panel the mean value is displayed in blue, the 2.5 percentile in green and the 97.5 percentile in red.

Check for any radical changes in the behaviour of the sensor over the course of the session. If this is the case it may be preferable to set up the amplitude adjustment separately for different parts of the session.[1]

In addition to these figures, the statistics are also stored in a text file (in the current example it would be named 'comppos_stats_ds_beststartl__comp6.txt').
This text file also includes the suggested range of rms, tangential velocity and euclidean distance for each sensor (calculated as the 5 and 95%ile over the upper and lower percentiles of the individual trials).
The essential information in this text file can be printed out in the command window with
`parsestats('comppos_stats_ds_beststartl__comp6.txt')`
The last five lines are the important ones, since they arrange the information in a way that can be directly used by the amplitude adjustment. Example:

```
rmsthreshb  = [  3     5     5     6     6     3     3     5    11     9     8     7];
velthreshb  = [ 170   150   230    40   110    40    60   220    30    40    80    90];
parameter7b = [ 1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0];
lolimb =      [75.0  70.0  76.5  67.0  96.0   NaN  69.5  89.0 155.0 154.5  83.0  85.5];
hilimb =      [97.0  95.0 100.5  70.0 110.0   NaN  76.0 115.0 157.0 159.5  92.5  96.0];
```

Sensors are arranged from left to right in each line.
The fives lines are as follows:

rmsthreshb: upper boundary for rms, i.e any data with rms above this value will be ignored in the regression analysis

velthreshb: upper boundary for tangential velocity

parameter7b: currently irrelevant for tapad-based data, but very important for kalman (see below)

lolimb: lower limit of euclidean distance from comparison sensor

hilimb: upper limit of euclidean distance from comparison sensor

(the last two lines have NaN for sensor 6, since the comparison sensor cannot be compared with itself)

It is important to realize that these statistics are basically dumb: if there is an unusually large proportion of unstable data for any sensor then they will not be realistic. For example, the expected range of euclidean distances for the tongue relative to the nose sensor is about 20mm. If the automatic statistics give a value of, say, 50mm, then one will have to refer back to the figures to see if a subset of potentially stable trials suggests a more plausible value.

A further important issue comes up here for the first time: The relative stability of the different reference sensors. Clearly, if reference sensors can be absolutely rigidly fixed to the head, and if measurement accuracy is perfect, then the distance between any pair of reference

[1] An additional set of figures is also generated, with names like `comppos_stats_ds_beststartl__comp6_trialn_1.fig`. These show the total and valid data per trial. If all data in the trial is valid then only one line will actually be visible. This information is not needed for preparing the amplitude adjustment, but it should be checked to make sure that there are no trials with unexpected lengths (and that no missing data (NaNs) have been overlooked).

sensors should not change. Thus, in the above statistics, the lower limit and upper limit of the euclidean distance should be very close together. Later, we will need to decide which reference sensors to use for normalization of head position. We here get a first indication of whether there are noticeable differences in the likely quality of the data for the different reference sensors.

Open the base version of the wrapper function do_ampvsposamp and save under this name. Fill in the locations with '??' with appropriate values. Paste in the last five lines from parsestats in the location indicated, and decide whether any of these threshold values need changing by hand. The main possible complication involves deciding whether to subdivide the session.

While the script is running there is a great deal of output in the command window and also various graphics are generated.
However, as the processing can take quite a while it is often more convenient to look at the output afterwards (note that separate sets of output files are generated for each sub-part). The graphics are stored in subdirectory 'figs/'.
The command window output is stored in a log file something like 'ampvsposampstats_dsbeststartl1_log.txt' (depending on the path to the input data, sub-part indicated by the digit before '_log').

In the log file the main thing to check is that unexpected amounts of data have not been excluded (this may mean the thresholds for rms etc. have been set wrong).
Examine the lines like:

```
Total/unreliable samples : 4473   255
```

The details are given in lines like:

```
Trial 374. Bad samples (rms, vel. euc. p7 overall : 0   0  35   0  35
```

This indicates that a total of 35 samples have been eliminated for this trial for this sensor, all based on the euclidean distance thresholds.

The analyses have to be carried out for every combination of sensor and transmitter (indicated by lines like : `1 6`)
The following lines give estimates of the distribution of the residuals before and after the adjustment for ones such combination

```
Histograms (per 1000) of residuals (old then new) for bins with upper edges:
     5    10    15    20    25    30   Inf

================================
         604         396           0           0           0           0           0
        1000           0           0           0           0           0           0
```

This indicates that before adjustment 60.4% of the residuals were in the region of 0-5, and 39.6% in the region 5-10. After adjustment 100% are below 5.

This information is also available from the graphics (names like 'ampvsposampstats_dsbeststartl1_S1_oldnewres.fig' where 'S1' etc. stands for the sensor). They show scatter plots of the old residuals and estimated new residuals. Improvements in the rms can be expected if the new residuals cluster more closely around 0 than the old ones.

The most important output from the procedure is a small mat file named something like ‘ampvsposampstats_dsbeststartl1.mat’ (i.e basically the same name as the log file). It contains the results of the regression analyses, and is needed as input for the next stage. Normally the user will not need to be concerned with this file directly, but may need to check that it is actually present (or has the expected name) if the next stage fails to work.

Applying the adjustments
The previous procedure has performed the regression analysis to capture any systematic pattern in the residuals, but has not actually generated a new set of amp data.
This is carried out by the script do_adjamps.
There is not usually much that needs setting up in this script if the standard locations for the old amps (input) and new amps (output) are used.
The division of a session into sub-parts should be copied from do_ampvsposamp. The only difference is that no trials should be excluded.
The main decision is whether to apply the amplitude adjustments to the full data or to the downsampled data. If no sensors have shown major problems up to now then the adjustment can be applied to the full data. Otherwise, the adjustments can be applied to the downsampled data (followed by a further run of do_tapad_ds) in order to get a quicker idea of how serious any remaining problems may be. The main reason for applying the amplitude adjustment is to reduce the chances of unstable solutions. So if any sensors have looked unstable then it is, of course, interesting to determine whether the amplitude adjustment has improved the stability before continuing.
Here we will assume that no particular problems have been observed. Applying the adjustment to the full data will result in the creation of a new subdirectory ampsfiltadj1/amps. These amp data will in turn provide the input for running tapad on the full data.
Since this step has usually been run with a tapad option that requires the best possible version of the kalman solution as additional input we will first consider the application of amplitude adjustment to the kalman data.

Amplitude adjustments: special considerations for kalman data
This is based on a pair of scripts equivalent to those just used for the tapad data:
do_ampvsposampk.m
do_adjampsk.m

To set up do_ampvsposampk it will first be necessary to run do_do_comppos with basepath set to the current version of the kalman data, e.g
basepath=['ampsfilt' pathchar 'kalmanus' pathchar 'rawposm' pathchar];
Make sure the diary file is not commented out.
Get the statistics with e.g
parsestats('comppos_stats_kalmanus0_m_comp6.txt')
and paste them into the script.
The same considerations of checking for plausibility apply as above (i.e cross check with the graphics if necessary).
There is one special new feature of the statistics that applies to kalman but not to tapad, which crops up here under the uninformative name of ‘parameter7'.
When the kalman filter is calculated it is run forwards and backwards in time over the data of each trial. Parameter7 contains the euclidean distance in mm between the two solutions. If the kalman algorithm is able to generate a robust model of the data then the solutions should be

virtually identical: differences < 0.1mm. Values of more than a millimeter or so have proved in practice to be excellent indicators of suboptimal solutions. This information is particularly valuable for the kalman approach, because it is in the nature of the kalman filter to give fairly smooth output even when the solution is not reliable. Thus the tangential velocities are less likely to provide an indication of troublesome data than with tapad where the typical behaviour is to jump abruptly between different solutions in cases of instability.
It has also emerged in practice that the amplitude adjustment (applied to kalman data) can give a substantial improvement in Parameter7, indicating that the adjusted amplitudes allow a more consistent model of the data to be obtained.

The other main difference between do_ampvsposamp (tapad) and do_ampvsposampk (kalman) is that the latter involves a preliminary stage in which the posamps are calculated (they are calculated automatically by tapad, but not by kalman, during position calculation).

do_adjampsk.m is virtually identical to the tapad version. It has simply been set up so that the new amps are stored in a separate subdirectory (normally 'ampsfiltadj1k/amps').

As soon as the new amps have been generated, then do_kalman should be run again. This simply requires setting basepath to the new subdirectory e.g basepath='ampsfiltadj1k'.

Since this new set of kalman position data has a special role to play in the next tapad run it is a good idea to use do_do_comppos to compare the kalman solution before and after the amplitude adjustment, just to make sure that it really is stable.
Set basepath to the new solution, altpath to the old solution, comment out compsensor. Uncomment diaryfile if it is important to document potential changes in the stability of any sensors from the old to the new version.

Tapad with full data

Assuming the kalman solution is satisfactory, it is possible to do the full run of tapad in what is referred to as recursive mode.

This means that existing position data is used to provide start positions[2] on a sample by sample basis. This contrasts with the way we have used start positions up to now, in which the start position provided by the user is just used for the first sample in the trial. After that, the start position for calculating sample n+1 simply consists of the positions just calculated for sample n.

What are the advantages of the recursive approach?
1. Since the start position for every individual sample is probably very close to the actual solution tapad is less likely to go off into outer space.
2. Also tapad will run faster because fewer iterations are needed to reach the solution.
3. And if tapad does go off into outer space then this increases the chances that the outliers will be restricted to isolated samples: with the non-recursive procedure there

[2]Whenever we refer to start positions we always mean all five coordinates (i.e including the orientation).

is always the danger that the algorithm will become 'locked' into the vicinity of a wrong solution, since the wrong solution is used as the start position for the next sample. Isolated outliers are generally easier to identify and fix than extensive mistrackings (see the topic "velocity repair").

Possible complications:
Assume the kalman solution is very poor for one sensor. Tapad is set up so that not all sensors have to be processed the same way. Accordingly, one could first run tapad as outlined above for all sensors except one. Then tapad can be run again for the missing sensor, using the same output files since any sensors already in the output files are not overwritten. For this missing sensor there are a number of possibilities:

1. Use a downsampled tapad solution (if available and of satisfactory quality) as the recursive input (it is upsampled by the program to match the full data rate).
2. Simply run the program non-recursively with a single start position
3. As a desperate measure: There is a pair of functions that allows the position of a given sensor to be predicted on a sample by sample basis from a group of other sensors. This prediction can then be used as the recursive input. See the help text for predictsensorfromothers.m and calcsensorfromothers.m for details.

footnote: beststartl
footnote: multiple adjustments
footnote: origin of parameter 7