Amplitude Adjustment

Background

See Hoole/Zierdt (2010) for motivation and background to the procedures implemented here, and explanation of the term 'residual' that is used frequently below (especially sections 20.3.1 and 20.3.2.1, pp. 340-345).

The basic concepts: When the position calculation algorithm determines a position estimate for a given sensor it uses the magnetic field equations to generate the amplitude measurements that would be expected for that position. These expected amplitudes (6 for each sensor, corresponding to the 6 transmitters) are stored in the posamps subdirectory within the directory where the position data itself is stored. In any real-life situation the best overall estimate of the sensor positions will not generate exactly the amplitudes that were actually measured. The differences between the measured amplitudes and the generated 'posamps' are referred to as the residuals. The so-called rms value stored together with the position data is (as the name indicates) the root mean square value over the 6 residuals for each sensor.

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.

In principal these procedures can be done separately and in parallel for data based on Tapad, Calcpos and Kalman (if available).

For simplicity, we will only consider the case of Tapad data here[1].

The steps in detail

After loading do_do_comppos into the editor:
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

[1] At IPS we have usually followed these procedures for Tapad and Kalman data in parallel. This is outlined in detail in the parallel document covering the relevant procedures for Kalman data. We have not actually experimented with parallel use of Calcpos data, although this would be basically straightforward. After generating the adjusted amplitudes it will simply be necessary to convert them from mat-files back to raw binary files so that Calcpos can use them as input data (see the function saveamp.m).

movements with no speech, and conversely, also those where the subject is neither moving nor speaking at all. The log file generated by the prompt program used at IPS for running the experiments is usually used for this purpose.

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.[2]

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:

---

[2] 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).

```
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 and Calcpos-based data, but very important for Kalman (outlined in the parallel document)

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 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 just to the full data or to both the full and the downsampled data.
Here, we will assume that the adjustments are applied to both versions of the data, for reasons that will become clearer in the next section (when combining Tapad with Kalman then other choices may be more convenient).
After applying the adjustments to the downsampled data, do_tapad_ds should be run again with the adjusted amplitudes as input.
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. In other words, do_do_comppos can be used to compare the position data before and after amplitude adjustment. In cases where there are big differences between the versions then one hopes, of course, that the adjusted version looks preferable (e.g velocity patterns are more typical of speech data).

In a worst case scenario, it may turn out that some data (maybe for just one sensor) is not yet acceptably stable. In principle, it is possible to go through multiple cycles of adjustment for individual sensors, but this has rarely resulted in radical improvements. It may then be better to check whether one of the alternative position calculation procedures (Kalman, Calcpos) gives better results for that sensor.

Applying the amplitude 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.

Tapad with full data

Assuming the downsampled Tapad data based on the adjusted amplitudes 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[3] on a sample by sample basis.
The existing position data in this case is the downsampled Tapad data (upsampled internally be the program to match the full data rate).
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 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”).

The wrapper script for this version of tapad is do_tapad_full_base.m
This will need editing so that the variable 'recursivestartfile' contains the path to the downsampled Tapad data to be used as the recursive input.

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

Possible complications:
Assume the downsampled Tapad 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. Simply run the program non-recursively with a single start position
2. 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.

Final word

Setting up the amplitude adjustments is one of the most time-consuming tasks in processing the data, since a lot of user interaction is required.
If one is absolutely sure that the data is stable then one could consider skipping the adjustments altogether.
The adjustments could also probably be skipped for any sensor where the rms value does not exceed about 5, since a low rms means that the regression analysis on which the amplitude adjustment is based is unlikely to find any systematic pattern in the residuals.
Nevertheless, being forced to examine the data and collect statistics on velocity, euclidean distances etc. helps to build up a feeling for typical patterns in the data, and makes it less likely that one will overlook screwy data when it does occur.