Getting started with position calculation

For the purpose of illustrating the basic work-flow we will start by running the position calculation without providing the algorithm with any start positions.
Normally this situation does not occur in practice since it should be possible to specify very roughly in advance at least what the sensor orientations are likely to be (and use this for on-line calculations while the experiment is running).
However, if one suspects that the preliminary estimate was completely wrong, then one may want to start from scratch again.
Usually, even without start-positions the algorithm will succeed in calculating some usable data for most sensors. This can be enough to bootstrap a further pass over the data, leading to further refinement of the start-positions.

Further preliminary remark:
Position calculation can be performed with two completely different algorithms that will be referred to as TAPAD (designed by Andreas Zierdt) and Kalman (implemented by Korin Richmond). As far as possible, we will use both in parallel since this provides the best cross-checks on the overall reliability of the data.
However, we will start with the Kalman version, simply because it is much faster.

Load the wrapper script template do_kalman_base into the editor and store it as do_kalman. Insert appropriate specifications for the current experiment wherever '??' appears in the template: These are the variables kalmansuffix and triallist. Set kalmansuffix to something like 'ns' to indicate "no start positions". Also, as with all these wrapper scripts include a brief comment indicating the experiment it is designed to process, and include a list of the sensors for reference (see the comments in the script). The variable 'chanlist' will also need editing if not all sensors are actually in use for the group of trials that you want to process.

Start the processing with do_kalman, wait for a few trials to be processed to estimate how long the processing will take, and then decide whether to go for a cup of coffee in the mean time.

Important note: The Kalman version of the position calculation can only be run on Linux machines (if you normally work on a Windows machine you can connect to a Linux machine using ssh, and run matlab there via the ssh connection).

While the calculations are proceeding various figures are output on the screen (for each trial, one line per sensor). This gives some information on the expected quality of the solution, but usually flashes past to fast too be useful. See the help text for kalman_fb.m for details.

During processing, several new subdirectories are created below the kalman subdirectory. The only one we will be concerned with is 'rawposm'.

Assessing the results

Several procedures are available for numerically and graphically assessing the quality of the results. These leads to a new wrapper script 'do_do_comppos.m'. We will start with a very basic version and gradually add to it as we move through each processing stage.

Run the script with ‘doshowtrial’ set to 1 (and after replacing ‘??’ in the template script with appropriate values.

Find the figure that shows values for position, orientation, rms, tangential velocity and nan-count over trials.

Try and determine a range of trials that give stable results, and find out if any sensors have particular problems.
To help with this use the program ‘showstats’. This gives a numeric summary of results for selected trials or channels. Use its mode 1 to check the relative position of the sensors makes sense.
When a group of stable trials has been identified, run showstats with mode 7. You will be prompted to give a filename in which start positions will be stored.

Then go back to do_kalman and set it up to run again, now using the start positions.
Set kalmansuffix to a new value, and enter the name of the file with the start positions.

Run the kalman procedure again, and examine the results as before. If the results look more stable than those from the preliminary pass, then consider recomputing the start positions and even running the kalman procedure yet again. At this stage it may even be worth computing different sets of start positions for different parts of a session if the position of the subject in the field has changed substantially. It will then be necessary to expand the do_kalman script accordingly.

Once one has converged on a basically stable set of start positions, then they can be also used when running the TAPAD algorithm.
Load the file do_tapad_ds_base.m and save as do_tapad_ds.m. This is basically very similar to the do_kalman script. Set the appropriate values for ‘startfile’ and ‘triallist’. Otherwise, nothing should need changing.
Run the procedure. It will take about 50 times longer than the kalman version, even though we are using a downsampled version of the data.

Check the results of the tapad version as done previously for the kalman version.
This will mean setting the path in do_do_comppos to

```
basepath=['ampsfiltds' pathchar 'beststartl' pathchar 'rawpos'
pathchar];
```

If there is any suspicion that either the tapad or kalman solution has instabilities then do_do_comppos can be used to compare them.
Set doshowtrial to 0. Set basepath to point to the kalman solution, and set altpath to point to the

tapad solution. This displays the two solutions in parallel. The distance between the two solutions is given in the panel labelled ‘Euclidean distance’. If the solutions ever diverge widely, i.e by more than a millimeter or two, then one can try and make a preliminary decision as to which version is more plausible. More details on this display mode of do_do_comppos are given later. [1]

---

[1] Since in this example downsampled data (tapad) is being compared with full data (Kalman) an absolutely perfect match is not to be expected, particularly in regions where movement velocity is high.