Final steps

Assuming one has been successful in generating satisfactory position data with Tapad, and perhaps also with one of the additional procedures (Kalman or Calcpos), there remain two further steps whose main purpose is to ensure that any remaining problems are not overlooked.

1. Velocity repair
This is now something of a misnomer, since the repair function should only be used very rarely.
Some background to this feature is given below.

Currently, the more relevant aspect of this processing step is that it filters the position data again with the filters that were used at the start of processing to filter the raw amplitudes.

Filtering
The input to this stage is the output data from the full Tapad calculation (this will usually have 'recurse' as part of the path name)

Wrapper script: do_velocityrepair_base.m

The data generated by this step (stored in files with 'velrep' in the path name) represents the final stage of the basic position calculation.

Why would one want to filter the data again? After all, the amplitudes had been filtered right at the beginning of processing. There are two main reasons:
(1)
The non-linear optimization used by Tapad (and perhaps even more noticeably by Calcpos) may introduce some 'noise' (i.e slight fluctuation in the solution, but not as bad as a major discontinuity) not present in the input data. Low-pass filtering can be beneficial here.
(2)
Assume a situation in which some discontinuities in the position data had gone unnoticed. After filtering, the data may look plausible. But an important part of velocityrepair is that it does not just filter the data but also recalculates the posamps and rms for the filtered data. If one inadvertently smooths over a discontinuity then the position data immediately before and after the discontinuity will be highly inconsistent with the input amplitudes, resulting in a sharp increase in rms in this region. This increased rms should give a very obvious indication that there may be problems remaining in the data.

Repair
The background is in Hoole/Zierdt (2010), section 20.3.2.2 (pp. 345-347).
Essentially, 'repair' here means that a rather primitive algorithm is used to ensure continuity in the position data, given that Tapad (and Calcpos) have no built-in continuity constraints. It should be used very sparingly, as it only works well for eliminating very short and obvious discontinuities of just a couple of samples. Currently, it will generally be preferable to use the Kalman approach, if available, to replace the data of problematic sensors - maybe even just for one or two trials (see also Further Notes (2), below).

2. Euclidean distance between solutions

A good quality-control procedure is to store the difference between alternative solutions (i.e. it corresponds to the Euclidean distance shown by do_do_comppos when both basepath and altpath are defined). The best choice is to compare the Tapad solution (at the output of the velocity repair procedure) with the Kalman solution, if available. This combination is particularly attractive because Tapad and Kalman use fundamentally different algorithms. Thus, if their results are very similar, then this is a good indication that the solution is robust. If the results are ***not*** similar, then all is not lost (one of the two solutions may be fine), but one will have to devote some extra effort to deciding which of the solutions, if any, can be used.

The Euclidean distance is stored in the first set of input data (usually the Tapad solution) as the so-called ‘parameter 7' (and will be displayed in the corresponding panel of do_do_comppos).

Two possible alternatives if Kalman is unavailable:
(1) Simply compare the input and the output of the velocity repair
(this will give a similar indication to the increased rms in the vicinity of any discontinuity discussed under velocity repair above. i.e there will be a particularly big difference between filtered (smoothed) and unfiltered data in such a vicinity)
(2) Compare Tapad and Calcpos if the latter appears stable

Wrapper script: do_eucdist2pos_base.m

Further notes

(1)
Velocity repair (even just filtering) is probably superfluous for Kalman

(2)
Sometimes the Tapad solution may be basically OK, but Kalman (or perhaps Calcpos) is much better just for one individual sensor.
The following wrapper script can be used to replace the data from the ‘main’ solution with data from an alternative solution:
do_movesensor_base.m