Interactive comment on “tobac v1.0: towards a flexible framework for tracking and analysis of clouds in diverse datasets”
by Max Heikenfeld et al.

Anonymous Referee #1
Received and published: 25 June 2019

We would like to thank the reviewer for their comments and suggestions. The review pointed out important aspects that we have clarified or addressed by minor changes in the manuscript. In the following, we respond to the reviewer’s comments in black, with our answers to the comments in blue and the adapted text from the revised manuscript in green. We have attached the revised version of the manuscript with tracked changes to the general authors' response. In the general authors’ response (AR), we have added a few additional comments regarding the revised manuscript and points raised by both reviewers.

This paper present a tracking code for clouds and other atmospheric objects, using modern programming techniques and languages. The topic is certainly not new, but the fact that the code is developed to be flexible and openly available warrants a code description paper. I find the paper clearly written, appreciate the examples, and recommend publication after the following minor questions are answered:

1) Was it a conscious decision to not tack on “. . . and COnvection” to the acronym?

We actually played with the idea of that rather obvious acronym. But apart from avoiding possible negative associations with the health impacts of the associated substance, we decided to stick to a unique name for the package to make it easier in terms of package managers, documentation, searchability, etc. and thus opted for the acronym *tobac.*

2) What is the performance of this algorithm? For example, for each of the test cases, what was the maximum memory footprint, runtime of the script, etc.

We want to thank the reviewer specifically for this comment on a very important aspect of the software package, that we had not addressed quantitatively in the initial version of the manuscript. In fact, creating the detailed performance analyses for the example cases and some puzzling results lead us to more detailed profiling that, especially in the code for the feature detection, allowed to reduce the processing time by more than an order of magnitude. These changes were related to time-consuming operations related to creating specific data structures or writing values into these data structures and do not change the analysis results at all. As part of the testing and the improvements, we have modularised the code for the feature detection and segmentation into several sub-functions, which makes the code more accessible and also more suitable for future adaptions and additions.

We have added information about the typical performance for the examples in the manuscript text, taking both calculations on a typical laptop and on a dedicated data analysis facility (JASMIN) into account:

“The data processing has been performed on the JASMIN data analysis facility (CEDA, 2019). The script including feature detection, segmentation, trajectory linking and saving of the analysis output for the 1-min data output had a processing time of around 17 minutes with a maximum memory footprint of 3.1 GB using a maximum of 3 processes and 27

threads. The segmentation step has been broken up into chunks of 10 minutes each to limit the total memory consumption of the analysis. The processing time is almost entirely taken up by the segmentation step using time-resolved 3-dimensional data of the total condensate. It is thus strongly affected by the time required to access the data on the disk and thus highly dependent on both the infrastructure and the structure of the data file, i.e. the data compression in the input files."
(page 14, line 10)

"The data processing was performed on the JASMIN data analysis facility (CEDA, 2019). The total processing time of the script including feature detection, segmentation trajectory linking and saving the output data was around 1 minute with a maximum memory footprint of around 500 MB for the model data and 2.5 minutes with a memory footprint of around 400 MB for the satellite data, each using a maximum of 3 processes and 27 threads."
(page 18, line 28)

3) It makes sense to separate the several stages of the workflow from a coding point of view; is it also possible to save data after the trajectory linking, and if so, in what format and what kind of design?

The individual stages of the workflow are separated as discussed in the "software description" section. Intermediate results can be saved after the initial steps in the format of *pandas* DataFrames for features and cloud tracks or as *iris* cubes in the case of the cloud masks resulting from the segmentation step. We have added that explicit information in the "software description" section of the manuscript:

"The intermediate results of each individual analysis step can be conveniently saved and examined in the form of pandas DataFrames or iris cubes."
(page 6, line 23)

4) P6: Do I understand correctly that the feature detection only works in 2D? Dawe and Austin (and others) did 3D feature detection, which ends up being more precise, especially also in the tracking part.

The feature detection we have currently implemented only works in 2D. This was a reasonable choice, especially with regard to the deep convective clouds we mainly focused on here. As you mention, there will be situations such as the analyses of shallow convection you referred to, where 3D identification and 3D tracking will be beneficial. We have concrete plans to include 3D feature detection and tracking in the next major version of tobac.

5) P9, l20: A circular range parameter v_max suggests that the trajectory linking is performed isotropically. Is it possible to assign/automatically retrieve an advection velocity as well? Heiblum et al did this very effectively.

The advection velocity of the trajectories in previous time steps is actually used as part of the tracking using `trackpy`. Thus, v_max only describes the size of a circle around the predicted position based on the advection in the previous steps, i.e. the allowed deviation of from the predicted position using this information from the previous time steps. We have adapted the manuscript in this section to make this clearer and added a schematic depiction of a specific linking step as Figure 3.

“The linking determines which of the features detected in a specific time step (see Sect. 2.2) is identical to an existing feature in the previous time step and is illustrated in Fig. 3. For each existing feature, the movement within a time step is predicted based on the velocities in a number of previous time steps. The algorithm then breaks the search process down to candidate features by restricting the search to a circular search region centred around the predicted position of the feature in the next time step. For newly initialised trajectories, where no velocity from previous time steps is available, the algorithm resorts to the average velocity of the nearest tracked objects. The parameter v max restricts how much the future position of a feature is allowed to deviate from a linear extrapolation of the trajectory over time. It thus has the units of a velocity and describes the dependency of the circular search range d on the output time step Δt in the data used for the tracking *d = v max Δt*.”
(page 9, line 31)

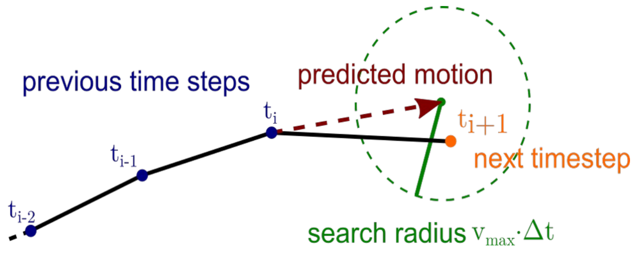

Figure 3. Schematic illustration of the trajectory linking with the predicted motion of the feature based on previous time steps and a search
range around the predicted position.
(page 10)