Dear Jeff Dozier,

Thanks for your interests in our manuscript and your inputs. Please find below our responses (in **black**) to your comments (in **blue**):

Hello. Interesting paper with the ray-tracing algorithm. Two comments:

(1) Recent paper Dozier (2022) parallelizes the original Dozier et al. (1981) codes and considers Earth curvature. The parallel implementation can be in the rotation angles or along the columns of the rotated grid. Computing horizons for a single azimuth takes only a few seconds on a grid about 2,000 x 3,000 cells. Obviously, the clock speed depends on the number of processors available. Generally, one gains more performance by parallelizing the rotation angles, but the maximum number of cores that can be brought to use is N/2, where N is the number of horizon azimuths. The speed is independent of the search radius, but of course there are edge effects: the computer cannot see beyond the edge of the grid. I recently processed a lidar DEM for 33,000 x 13,000 cells.

We set up a comparison of our ray casting based algorithm and the one you provide in MATLAB but encountered some issues, which we discuss in the following. Similar to Fig. 7 from our manuscript, we compared both algorithm for two DEM domains with various sizes and centred at latitude/longitude (46.5851°, 7.9607°) and (47.37174°, 8.54226°). We directly applied NASADEM data (without any coordinate transformation to rotated latitude/longitude coordinates as described in the manuscript) and run both algorithms in a single thread mode (i.e. without using parallelisation). We tried to run your algorithm with the following settings:

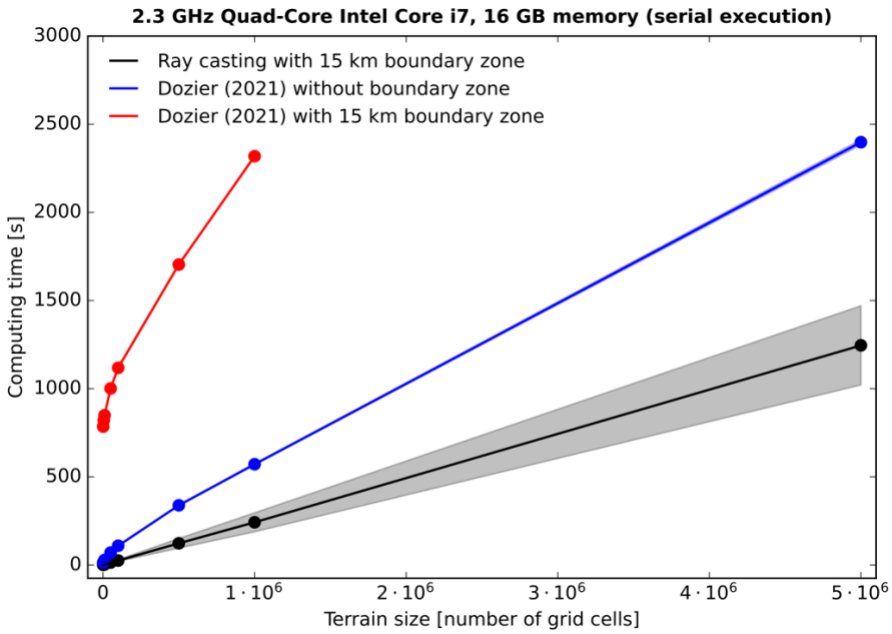
```
useParallel = false;
nHorz = 360;
[A,H] = horizonAllDirections(double(Z),R,'nHorz',nHorz);
```

However, this yielded the following error:

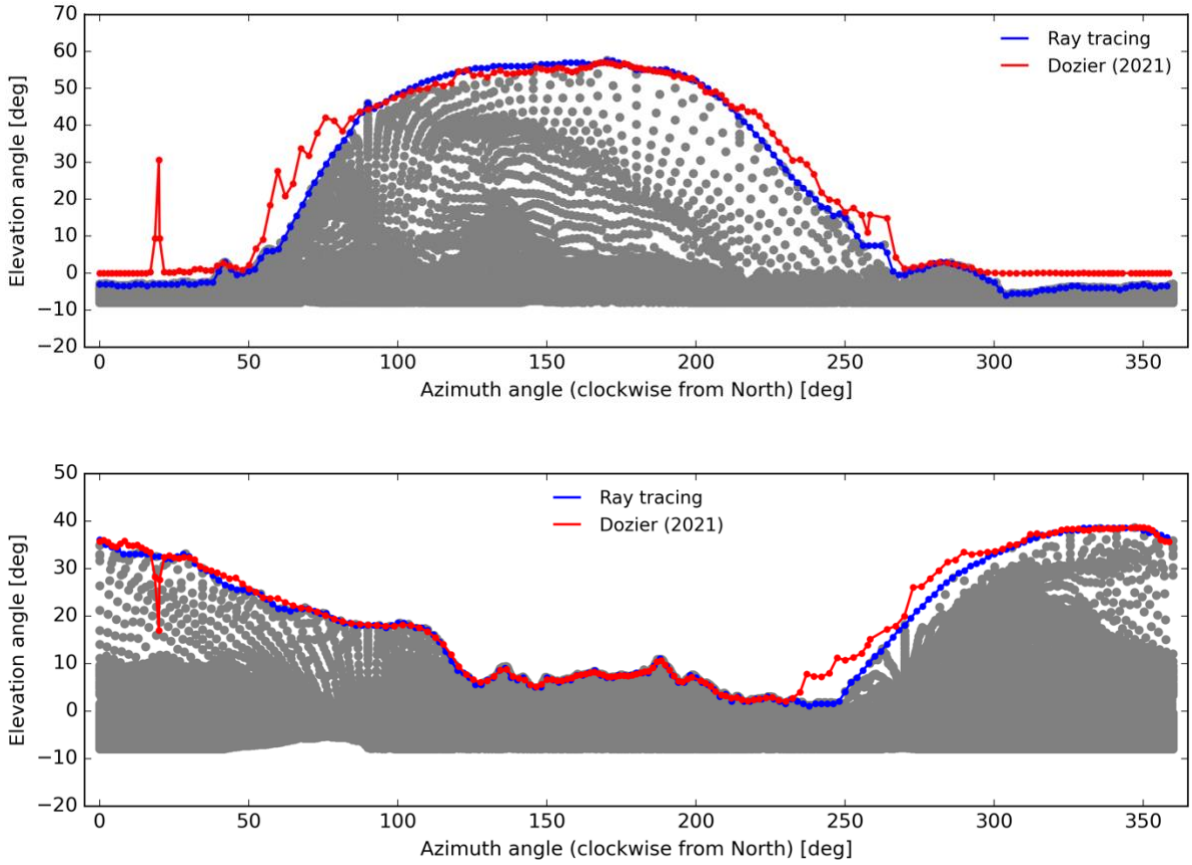
```
Error using smooth3 (line 65)
All elements of SIZE must be odd integers greater than or equal to 1.

Error in horizonAllDirections (line 248)
H = smooth3(H,'gaussian',kernelSize);
```

We then reduced the azimuth sampling number to 180, which allowed us to run the algorithm successfully. The performance evaluation is shown in the following figure:



The gray range for the ray casting algorithm stems from the slight terrain complexity dependence of the algorithm: the algorithm is faster for the domain centred at (47.37174°, 8.54226°), which represents a hilly landscape, and slightly slower for the domain centred at (46.5851°, 7.9607°), which represents extremely rugged and complex terrain. Compared to a “conventional horizon algorithms”, like e.g. Buzzi (2008) or Pillot et al. (2016), your algorithm is also substantially faster. Compared to our ray casting based approach, your algorithm needs approximately twice as long to process a DEM tile of a certain size. The difference in performance increases further in case a 15 km wide boundary zone is considered in your algorithm (to prevent an underestimation of horizon angles near the edges). The ray casting algorithm was also run with the same boundary zone (and a potential extension of this zone has a negligible influence on performance, as discussed in the manuscript). We were not able to run your algorithm for a terrain size of $5 * 10^6$ grid cells and a 15 km wide boundary zone on our test machine because the memory requirements exceeded 16 GB. Monitoring the memory demand of the algorithm during execution relvead a steady increase – this might be related to a memory leak in your MATLAB code.
In a second step, we compared the accuracy of both algorithms. The terrain horizon for two example locations, computed from an experiment with 1000*1000 grid cells and centred at (46.5851°, 7.9607°), is illustrated below:



The grey dots represent NASADEM elevation as a reference. This information was transformed to local spherical coordinates in the following order: Geodetic coordinates → Cartesian ECEF coordinates → local ENU coordinates (at the observer’s location) → spherical coordinates (i.e. all points were projected on a unit sphere centred at the observer’s location). We detected some artefacts in your algorithm, which appear most pronounced at an azimuth angle of ~20° in the above two locations. We found similar artefacts for other *nHorz*-settings but we did not investigate this issue in more detail. It might be related to an erroneous gap-filing / interpolation for certain DEM grid cells and azimuth directions in the algorithm. Some smaller deviations are also evident for other azimuth

directions from the above figure – these might be also related to gap filling / interpolation or to the “discontinuous” representation of the DEM in a raster (we discuss this issue in Sect. 4.2 of our manuscript).
Finally, we now also briefly dicuss the parallel scalability of our algorithm in the manuscript (see inset panel in Fig. 7a) as you do in Dozier (2021). For a domain size of 10^6 grid cells, employing 32 threads on a processor with 32 pyhsical cores yields a speed-up factor of ~27 compared to a single-thread execution.

(2) In the upslope direction, the calculation of the SVF must consider that the horizon could be the slope itself. See Equation (2) in Dozier (2022). Your method may take care of this problem, but you should check.

Thanks for pointing out this relevant detail. We consider this fact on line 258 (unrevised manuscript), where we define the relevant elevation angle for the SVF as the maximum of **(I)** the terrain horizon and **(II)** the intersection of the terrain surface plane with the unit sphere. In fact, our derived Eq. (10) can, by considering the different coordinate systems and applying multiple trigonometric identities, be rearranged to the SVF equation you presented in Dozier and Frew (1990).

Dozier, J., Bruno, J., & Downey, P. (1981). A faster solution to the horizon problem. Computers and Geosciences, 7, 145-151. https://doi.org/10.1016/0098-3004(81)90026-1

Dozier, J. (2022). Revisiting topographic horizons in the era of big data and parallel computing. IEEE Geoscience and Remote Sensing Letters, 19, 8024605. https://doi.org/10.1109/LGRS.2021.3125278 [published open-access]

Code is available: (https://www.mathworks.com/matlabcentral/fileexchange/94800-topographic-horizons)

With best regards,

Christian R. Steger, Benjamin Steger and Christoph Schär