Large-Window Curvature Computations for High-Resolution Digital Elevation Models

Anne M. Denton, *Member, IEEE*, Rahul Gomes, *Member, IEEE*, David M. Schwartz, and David W. Franzen

***Abstract*— With the increasing availability of high-resolution digital elevation model (DEM) data, a need has emerged for new processing techniques. Topographic variables, such as slope and curvature, are relevant on length scales far larger than the pixel resolution of modern DEM datasets. An approach for computing slope and curvature is proposed that uses standard regression coefficients over large windows while generating output on the full resolution of the original data, without adding substantially to the computation time. In the proposed window-aggregation approach, aggregates for fitting a quadratic function are computed iteratively from the DEM data in a process that scales logarithmically with the window size. It is shown that the window-aggregation algorithm produces the results of much higher quality than the two-step process of applying neighborhood operations such as focal statistics followed by small-window topographic computations, at comparable computational cost.**

***Index Terms*— Computational infrastructure and geographic information system (GIS), light detection and ranging (LiDAR) data, surface and subsurface properties.**

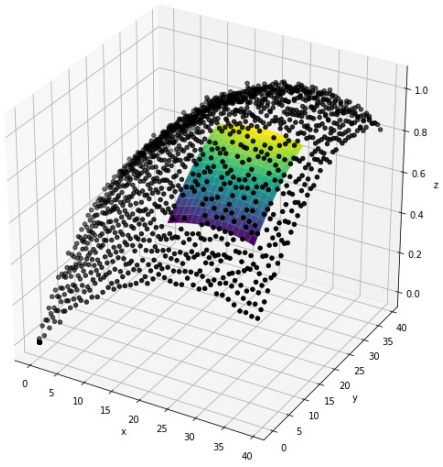

Fig. 1. Schematic of a large window fit to a noisy DEM.

I. Introduction

AIRBORNE light detection and ranging (LiDAR) technology has resulted in an abundance of high-resolution digital elevation models (DEMs) with resolutions of 1 m or less [1]. Applications of DEMs include understanding surface hydrology [2], [3], [4], [5] and soil erosion [6], [7] and performing surface segmentation [8], [9]. Many topographic variables can be derived from DEMs, with slope and curvature among the most important ones [10], [11], [12], [13], [14], [15].

It is assumed throughout this article that the extraction of a digital topographic model from LiDAR measurements has been completed and a DEM has been constructed. Inference of topography from vector data, including the removal of surface vegetation and built objects [16], is in itself a challenging topic that has recently gained in importance. Artificial intelligence approaches are commonly used to improve reconstruction quality [17], [18] and model vegetation [19], [20]. The use of curvature and elevation together has been shown to help in terrain reconstruction [21], [22], [23]. The presented work assumes that any such steps have been completed.

The history of the development of slope and curvature variables has been dominated by the use of 3×3 pixel windows [24], and the corresponding algorithms are most widely available in geographic information systems (GISs), such as ArcGIS [25]. Such window sizes are appropriate when the raster resolution is already on a length scale that is geomorphometrically relevant, i.e., the curvature corresponds to processes that relate to the formation of the landscape. For 1-m DEMs, it is not unlikely that variations between neighboring raster points are due to imperfect processing of the DEM. This type of high-frequency noise is a relatively recent problem, and until the advent of LiDAR, too low resolution of DEMs was a much bigger concern.

For high-resolution images, the appropriate window size can be easily 64×64 or higher. By default, an algorithm that inspects all raster points in such a window would require accessing more than 4000 elevation values for each output curvature value. This requirement creates computational complexity issues, even when common regression-based definitions of curvature can be naturally generalized to large windows. Already in 1980, Evans [26] proposed that curvature could be defined based on regression. At the time, he used this definition for driving coefficients over 3×3 windows. Wood [27], [28] later proposed a model of curvature that is applicable to larger windows, but the scope was limited to windows

Manuscript received 12 December 2021; revised 20 May 2022; accepted 2 July 2022. Date of publication 19 August 2022; date of current version 27 September 2022. This work was supported by the National Science Foundation under Grant IIA-1355466. *(Corresponding author: Anne M. Denton.)*

Anne M. Denton and David M. Schwartz are with the Department of Computer Science, North Dakota State University, Fargo ND 58102 USA (e-mail: anne.denton@ndsu.edu).

Rahul Gomes is with the Department of Computer Science, University of Wisconsin–Eau Claire, Eau Claire, WI 54701 USA.

David W. Franzen is with the School of Natural Resource Sciences, North Dakota State University, Fargo ND 54701 USA.

Digital Object Identifier 10.1109/TGRS.2022.3200354

of sizes not much larger than 5×5 or 7×7. The ability to increase window size for working across multiple window scales and for reducing the impact of errors in DEMs is well documented [29]. However, Wood's solution would not be practical for windows with thousands of data points, and with the DEM resolution in the 1990s, there was little motivation toward using such large windows.

Other than performance considerations, there is nothing fundamental that limits the use of regression-based curvature definitions to small-window sizes. Fig. 1 shows the problem. Elevation data are treated as sets of noisy measurements that depend on 2-D spatial coordinates. In statistics, quadratic functions are fit routinely to large datasets. The curvature problem is only special in the sense that this same computation, conceptually, is needed for every output raster point in the DEM. Performing this computation by iterating over all points in each window separately would be prohibitively slow. The problem can be made computationally tractable by recognizing that polynomial regression requires nothing but additive functions, i.e., functions for which all final aggregates can be produced by aggregating previous aggregates. This property enables a massive reuse of intermediate results between shifted windows.

The contribution of this work is to make computationally feasible a window-based regression task that was defined by Evans four decades ago, but for which the need for applying it to large windows is driven by recent increases in the DEM resolution. The regression does not involve an approximation per se, but there are some constraints on the proposed window-aggregation algorithm that could be viewed as approximations. Specifically, windows are expected to be square and their size to be a power of 2. The discussion is limited to geographical regions small enough that Earth's spherical shape can be ignored and the DEM can be treated as defined within a plane in the Euclidean space. Existing implementations of regression-based curvature measures are only available for windows with a small fixed number raster points. The *de facto* workaround of preprocessing the data by using a focal statistics tool for smoothing [25], [30] and then applying a small-window definition of curvature is not equivalent to the original regression problem and can hence be viewed as an approximation. Section IV shows that, in the presence of noise, this two-step approach results in a far noisier output than regression-based curvature. The latter has the added benefit that simple linear regression is arguably one of the best-understood tools in statistics. Minár *et al.* [31] highlighted the importance of using well-understood measures toward achieving exactness in modeling.

Another approach for working with high-resolution data is to use downsampling for converting it to a scale at which 3×3 windows are useful. Such an approach largely misses out on the promise of higher resolution, although it may help alleviate accuracy concerns that have been discussed extensively [32], [33], [34], [35]. The importance of scale is well known in the modeling of topographic data [36], [37], [38], [39], and problems related to 3×3 windows have been discussed extensively [35], [40], [41], [42], [43], [44], [45]. Mitášová and Hofierka [46] used spline-based interpolation to achieve an independence of the raster scale, and triangulated irregular networks have been used for computing curvature without being tied to a particular grid [33].

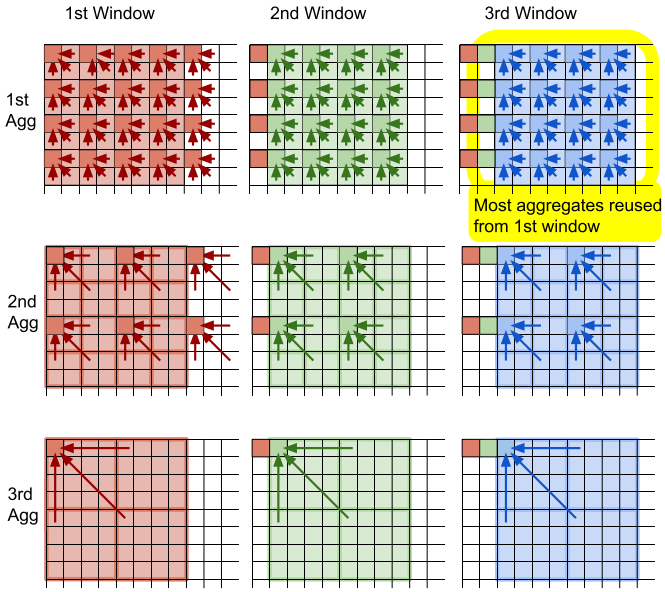

Fig. 2. Process for aggregating sliding windows shown for the first three 8×8 windows in DEM: first iteration (top row) produces 2×2 subwindows, second iteration produces 4×4 subwindows, and third iteration (bottom row) produces 8×8 subwindows.

The proposed approach uses a conventional mapping from an input grid of square raster points to an output grid of the same resolution. It also assumes that the Earth's surface can be considered as a plane within a Euclidean vector space. Earth's curvature, which is important for low-resolution DEMs [47], [48] and creates significant challenges when integrating data from multiple sources [49], is not the focus of the types of data analysis considered, as can be seen from a simple estimate: At 1-m resolution, a 64×64 window, which is the largest considered in this article, only has a spatial extent of 64 m in both dimensions. For conventional Landsat data, with its 30-m resolution, even a window as small as 3×3 raster points would have a larger spatial extent (90 m in both dimensions).

Fig. 2 shows the aggregation process that underlies the window-aggregation algorithm for the first three windows of size 8×8. The top row corresponds to the first iteration over the dataset. Aggregation is done over all sliding windows of size 2×2. The aggregates represent the middle point in the 2×2 window, which is not a raster point in the original raster. This shift by half a pixel is accounted for through adjustments in the metadata. Since the windows in the window-aggregation algorithm all have sizes that are powers of 2, there is no central pixel that is aligned with the original raster for any of the window sizes. All the results are treated as shifted by $(\omega - 1)/2$ in both dimensions, and the shift is reflected in the metadata of the file. Processing the result such that the raster matches with the original data requires an additional reraster step. Note also that each image is smaller by $\omega - 1$ pixels than the original one, i.e., there is a frame of width $(\omega - 1)/2$ around each of the output maps.

In ArcGIS, an estimate of the periphery is returned even when only incomplete information is available. For the window-aggregation algorithm, it would be straightforward to fill the periphery of large-window estimates by small-window ones, but it was not done in this work for the sake of clarity. Note that peripheral raster points were discarded for the comparison algorithm as well. Not doing so would have amounted to an unfair comparison since the peripheral points were of much lower quality than central points.

The next iteration is shown in the middle row of Fig. 2. In this step, windows of size 4×4 are produced from their four quadrants. Note that the aggregates are not retrieved from adjacent locations, but rather from locations that are $\delta = 2$ pixels apart. In general, the aggregation is done on windows that are $\delta = w/2$ raster points apart, where w is the current window size, which is twice w of the previous iteration $w_{\text{current}} = 2\delta = 2\ w_{\text{previous}}$. Not all of the 2×2 aggregates that are within the area of the red window are used for computing the 8×8 aggregates of that window. In fact, none of the 2×2 aggregates that are useful for the evaluation of 8×8 aggregates of the first (red) window contribute to the final aggregate of the second (green) window, as shown in Fig. 2. Reuse becomes evident between the first and the third window. From the top row, it can be seen that the first and third windows share 12 out of the 16 2×2 aggregates that contribute to the final 8×8 window. The final iteration is shown in the bottom row. In this iteration, 8×8 windows are constructed from their constituent 4×4 quadrants. For the small example, reuse cannot be seen at this level, but away from the borders of an image, any aggregate is reused four times and itself uses aggregates that were reused four times. It is this compounding of reuse that enables the logarithmic scaling of the algorithm.

The doubling of the window size in each iteration means that after n aggregation steps, the size of the window is $w = 2^n$. Since every aggregation step takes approximately the same time, the computation time can be estimated by solving for n, with the result being $n = \log_2(w)$. The scaling of the algorithm with regard to w is hence $O(\log(w))$, implying that working with windows of size $64 \times 64 (= 4096)$ pixels takes only twice as long as working with $8 \times 8 (= 64)$ pixel windows. The default brute-force approach of scanning each of the 4096 pixels in the window would take 64 times as long. The complexity of the brute-force approach is quadratic in the window size ($O(w^2)$), which is prohibitive for large window sizes.

In Section IV-C, it will be seen that, in practice, the computational effort is comparable to the combined complexity of preprocessing and computing curvature over DEMs in conventional approaches that produce approximate solutions of much poorer quality. Meanwhile, the result of the window-aggregation algorithm is a standard statistical fit to all data with no ad hoc assumptions. Moreover, it is in the nature of the algorithm that intermediate aggregates are computed, which can be used for multiscalar analysis.

The iterative aggregation that enables the efficient processing in this article has been previously used for calculating correlations between attributes in multiple raster layers [50]. In previous work, Denton *et al.* [51] demonstrated that spatial variables can be aggregated toward slope-related computations, an approach that is generalized to curvature in this work. The potential of using regression-based mean curvature toward prediction tasks was illustrated in [52].

A summary of curvatures and other topographic variables is given by Schmidt *et al.* [11] and a visual interpretation was done by Florinsky [13]. In GIS, the most important curvatures are ones that are defined with regard to slope. Their practical relevance stems from the force that gravity exerts on flows. In this article, the focus is on profile curvature, which is the curvature in the direction of the slope, based on the definition proposed by Evans [10], and tangential curvature, which is the curvature perpendicular to slope [53]. Concepts of the window-aggregation algorithm could be equally used for the mathematically interesting maximum, minimum, mean, and Gaussian curvatures, but those measures tend to be of less interest in practical applications.

Also of interest is the contour curvature or the curvature of the contour lines within a 2-D plane, which is historically called "plan" or "planform curvature." However, tangential curvature is a closer approximation than contour curvature to the ArcGIS planform curvature, which is consistent with the observation of Schmidt *et al.* [11] that the Arc/Info user documentation discusses "plan curvature" as implementing the equations for tangential curvature, given in [4] and [54]. Note that the fit function used by Zevenbergen and Thorne [54] includes nine terms instead of Evans's six terms, which allows it to fit a 3×3 window exactly. Zevenbergen and Thorne's equation includes terms that would be part of third and fourth orders in a Taylor series, and some differences between the presented results and ArcGIS's are hence expected. The challenges of relating published algorithms with actual implementations in systems is examined in detail by Blaga [12].

Slope and curvature are not the only types of derived attributes that can be computed using iterative aggregation of regression terms (IARTs) over sliding windows. Third-order coefficients could be derived in the same form [55] and some other properties that are commonly considered in geomorphometric analysis [56], such as variance of the local elevation distribution, may also be amenable to such processing. Likewise, it would be possible to define window-based slope of slope or slope of aspect measures, as they are discussed by Hu *et al.* [57]. Evans and Minár [58] introduced a taxonomy of curvature and other geomorphometric variables. While iterative aggregation can be potentially applied to both point- and area-based field variables, it is limited to ones, for which the aggregation can be done in multiple steps, i.e., all relevant aggregates have to be additive. Identification of further additive geomorphometric variables and their computation is left to future work.

II. Concepts

A. Basics of Window Aggregation

In the following, it is assumed that a high-resolution DEM in Universal Transverse Mercator (UTM) Projection is available, which allows considering the data as residing within a metric

grid in the Euclidean space. The elevation z in meters is effectively a dataset of samples of a potentially noisy function $z(x, y)$, for which easting $0 \leq x < M$ and northing $0 \leq y < N$ are also both given in meters. Notice that elevation datasets may apply a scaling factor to the z-value, which is left out of the original derivations for simplicity.

In a brute-force attempt, every possible window of a given size of $w \times w$ raster points would be extracted out of the overall dataset $z(x, y)$ and averaged as follows:

$$\left\langle z^{(w)}(x_0, y_0)\right\rangle = \frac{1}{w^2} \sum_{x,y \in \text{window at}(x_0,y_0)} z(x, y) \tag{1}$$

where z is the elevation, x_0 and y_0 are easting and northing, respectively, of the center of the window, and the notation $\langle \ldots \rangle$ signifies averaging. This aggregation would be performed over each window, one at a time, $M-w+1$ times in the x-direction (easting) and $N - w + 1$ times in the y-direction (northing). The prohibitiveness of this approach for large window sizes, which motivates deriving a way of rewriting the window-based means over z, will be derived, in which aggregates from prior iterations, corresponding to smaller window sizes, are used.

For simplicity, windows are indexed by their top-left corner, i.e., $z^{(w)}_{i_0 j_0}$ holds the aggregate for the window that has i_0 and j_0 as its smallest value of i and j, respectively. The shifting by $(w - 1)/2$ in each direction, which is required to place the derived raster image such that the values are at the center of the window they represent, is handled at the level of the metadata. With that convention, (1) becomes

$$\left\langle z\right\rangle^{(w)}_{i_0 j_0} = \frac{1}{w^2} \sum_{i=i_0}^{i_0+w-1} \sum_{j=j_0}^{j_0+w-1} z_{ij}. \tag{2}$$

Following the motivation in Section I, this sum is rewritten in terms of its four constituent quadrants, which are assumed to be available from the previous iteration at half the window size:

$$\begin{aligned}\left\langle z\right\rangle^{(w)}_{i_0 j_0} = &\frac{1}{w^2} \sum_{i=i_0}^{i_0+w/2-1} \sum_{j=j_0}^{j_0+w/2-1} z_{ij} \\ &+\frac{1}{w^2} \sum_{i=i_0+w/2}^{i_0+w-1} \sum_{j=j_0}^{j_0+w/2-1} z_{ij} \\ &+\frac{1}{w^2} \sum_{i=i_0}^{i_0+w/2-1} \sum_{j=j_0+w/2}^{j_0+w-1} z_{ij} \\ &+\frac{1}{w^2} \sum_{i=i_0+w/2}^{i_0+w-1} \sum_{j=j_0+w/2}^{j_0+w-1} z_{ij}.\end{aligned} \tag{3}$$

The rewriting is possible due to the additivity property of sums. In other words, for the summation, it is possible to compute an overall sum as a sum of partial sums. A recursive definition for $z^{(w)}_{i_0 j_0}$ follows by substituting the definition of $z^{(w')}_{ij}$ for $w' = w/2$:

$$\left\langle z\right\rangle^{(w)}_{i_0 j_0} = \begin{cases} \frac{1}{4}\left\langle z\right\rangle^{(w/2)}_{i_0 j_0} + \left\langle z\right\rangle^{(w/2)}_{(i_0+w/2) j_0} \\ +\frac{1}{4}\left\langle z\right\rangle^{(w/2)}_{i_0 (j_0+w/2)} \\ +\frac{1}{4}\left\langle z\right\rangle^{(w/2)}_{(i_0+w/2)(j_0+w/2)} & \text{for } w \geq 2 \\ z_{ij} \quad \text{for } w = 1. \end{cases} \tag{4}$$

Since windows cannot extend beyond image boundaries, the following limits apply: $i_0 < M - w + 1$ and $j_0 < N - w + 1$. The number of windows that are returned in each iteration decreases accordingly. Note that, although this recursive definition captures concisely how $\langle z\rangle$ at a particular value of w relate to those at $w/2$, a recursive implementation would not have the necessary performance. Instead, an iterative implementation is used, which computes averages one level at a time.

The derivation so far has been limited to averaging the elevation itself. Slope and curvature measures require computing spatial derivatives of elevation, which use aggregates that involve not only elevation but also the spatial coordinates. As such, the aggregation process has to explicitly represent x and y. In addition to $\langle z\rangle$, defined in (4), the following will be needed:

$$\begin{aligned}\left\langle xz\right\rangle^{(w)}_{i_0 j_0} &= \frac{1}{w^2} \sum_{i=i_0}^{i_0+w-1} x_i \sum_{j=j_0}^{j_0+w-1} z_{ij} \\ \left\langle yz\right\rangle^{(w)}_{i_0 j_0} &= \frac{1}{w^2} \sum_{j=j_0}^{j_0+w-1} y_j \sum_{i=i_0}^{i_0+w-1} z_{ij} \\ \left\langle x^2z\right\rangle^{(w)}_{i_0 j_0} &= \frac{1}{w^2} \sum_{i=i_0}^{i_0+w-1} x_i^2 \sum_{j=j_0}^{j_0+w-1} z_{ij} \\ \left\langle y^2z\right\rangle^{(w)}_{i_0 j_0} &= \frac{1}{w^2} \sum_{j=j_0}^{j_0+w-1} y_j^2 \sum_{i=i_0}^{i_0+w-1} z_{ij} \\ \left\langle xyz\right\rangle^{(w)}_{i_0 j_0} &= \frac{1}{w^2} \sum_{i=i_0}^{i_0+w-1} x_i \sum_{j=j_0}^{j_0+w-1} y_j z_{ij}.\end{aligned} \tag{5}$$

The derivation of these quantities will be the subject of Section II-D. First, the relationship between the geospatial derivatives and a least-squares fit of a quadratic function over an arbitrary window size will be discussed. Fig. 3 shows an overview of the steps in the algorithm and gives a preview of where the equations that will be derived in Sections II-B and II-C are needed. Section III provides more detailed pseudocode of the algorithm (Algorithm 1).

B. Geospatial Derivatives From Least-Squares Fitting

Following [10] and [28], geospatial derivatives are derived using least-squares fitting. However, unlike past approaches, a recursive window definition is used for the fit, which allows computing the coefficients, and thereby derivatives, for arbitrarily large windows at logarithmic computational cost. Before addressing these computational challenges in Sections II-C and II-D, the relationship between slope and

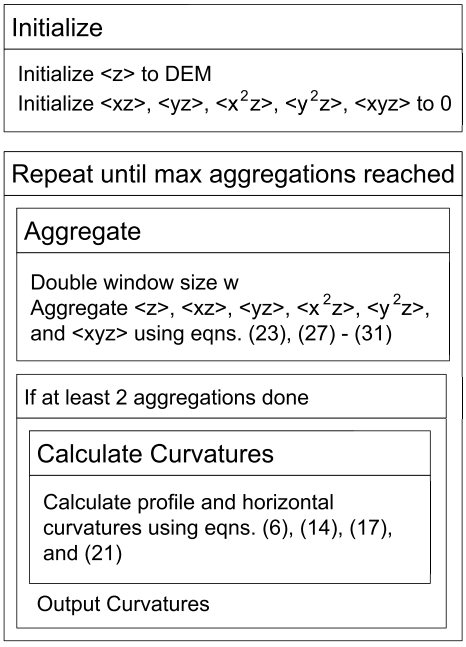

Fig. 3. Outline of the algorithm.

curvature on the one hand and linear and quadratic derivatives of a topographic surface on the other will be examined. The notation used in the following goes back to Shary [59] and is used in other publications on the topic as well [60]:

$$p = \frac{\partial z}{\partial x}, \quad q = \frac{\partial z}{\partial y}$$
$$r = \frac{\partial^2 z}{\partial x^2}, \quad t = \frac{\partial^2 z}{\partial y^2}, \quad s = \frac{\partial^2 z}{\partial x \partial y}. \tag{6}$$

With these definitions, any of the linear and quadratic topographic functions discussed, for example, by Schmidt *et al.* [11] can be computed. In the following, the focus will be on slope, profile (as proposed by Evans [10]), and tangential curvature (as proposed by Krcho [53]). The equations are given in the following [13], with a factor of 100 included that helps with discussing output:

$$G = \arctan\sqrt{p^2+q^2}$$
$$k_{\text{proper\_profile}} = -100\,\frac{p^2r + 2pqs + q^2t}{\left(p^2+q^2\right)\sqrt{\left(1+p^2+q^2\right)^3}}$$
$$k_{\text{proper\_tangential}} = -100\,\frac{q^2r - 2pqs + p^2t}{\left(p^2+q^2\right)\sqrt{1+p^2+q^2}}$$
$$k_{\text{contour}} = -100\,\frac{q^2r - 2pqs + p^2t}{\sqrt{\left(p^2+q^2\right)^3}} \tag{7}$$

where G is the slope, $k_{\text{proper\_profile}}$ is the curvature in the direction of the line of steepest descent, $k_{\text{prope\_tangential}}$ is the curvature within the plane that is perpendicular to the slope vector, and k_{contour} is the curvature of the contour line in a 2-D projection of the surface.

Differences between the curvature definitions are illustrated by Florinsky [13]. It can be seen there that contour curvature remains high even in regions with relatively low slope. Contour lines can have high curvature no matter how flat regions are, which is consistent with the mathematical representation that has an additional factor of $(p^2+q^2)^{1/2}$ in the denominator of the contour curvature, in comparison with tangential curvature, allows it to diverge for small slope. Tangential curvature, in contrast, has a factor of $(1+p^2+q^2)^{1/2}$, which does not tend to zero for small slope. The latter is somewhat comparable to ArcGIS's "plan curvature" as noted in [11].

Comparisons showed that none of these three curvatures is exactly what is implemented in ArcGIS. In fact, a simple test can be used to see that ArcGIS does not implement mathematical curvature. The mathematical curvature of a dome that is shaped like a section of a sphere is constant in any direction, i.e., both profile and planform are constant. Implementing the above formulas indeed produces this result. Stretching the dome upward by multiplying every raster point with a constant factor $c_h > 1$ results in an ellipsoid that has higher curvature in the middle than toward the periphery. In contrast, ArcGIS's curvatures are consistently larger toward the periphery of a spherical dome than the center, as would be expected for second derivatives. Moreover, the ratio of peripheral to central curvature does not change, regardless of the factor by which the dome is stretched upward. The following portions of the above formulas satisfy the second-derivative-like expectation that a constant factor leads to only quantitative but not qualitative changes in curvature, i.e., that $k_{\text{ArcGIS}}(c_h * f(x,y)) = c_h * k_{\text{ArcGIS}}(f(x,y))$:

$$k_{\text{profile}} = -100\,\frac{p^2r + 2pqs + q^2t}{p^2+q^2}$$
$$k_{\text{tangential}} = -100\,\frac{q^2r - 2pqs + p^2t}{p^2+q^2}. \tag{8}$$

Note that the sign of the profile definition is that of the theoretical definition, which is the negative of what is used in ArcGIS. The factors of $((1+p^2+q^2)^3)^{1/2}$ and $(1+p^2+q^2)^{1/2}$ from (7) are now absent since they cause the qualitative changes in behavior upon stretching a DEM. The formulas in (8) indeed match the definitions that ArcGIS uses best. This was established by calculating the mean average percentage error (MAPE) for the numerically computed curvatures using a 3×3 window versus the analytical results for the artificial landscape that is discussed in detail in Section IV-A. Table I shows the results of that comparison. Note that the artificial image is intentionally somewhat extreme, with elevation changes of over 350 m within a 500 m $\times$ 500 m area.

TABLE I
MAPE BETWEEN COMPUTED AND THEORETICAL CURVATURES

MAPE	ArcGIS Profile	ArcGIS Planform
Profile	**1.35%**	1276%
Proper Profile	234%	4377%
Tangential	940%	**1.02%**
Proper Tangential	1239%	44.6%
Contour	105%	99.0%

It can be seen that the MAPE of ArcGIS's profile is clearly smallest with regard to the simplified profile definition from (8), in particular 1.35%, and likewise ArcGIS's planform best matches the simplified tangential definition with an error of only 1.02%. MAPE values close to or above 100% are not considered meaningful since MAPE is not symmetric in the values that are being compared. The only other error small enough to consider is ArcGIS's planform in comparison with the proper tangential curvature from (7), which has an error of 44.6%, consistent with the earlier discussion that ArcGIS's planform is overall more tangential-like than contour-like. While implementations of all the above-listed curvatures are provided, the definitions in (8) will be used for the comparisons since they give the smallest errors for ArcGIS.

The next step is to examine how the quantities p, q, r, s, and t can be derived from a least-squares fit of a quadratic function. Generalizing the quadratic fit function $y^{(\text{quad})}(x) = ax^2+bx+c$ to two dimensions gives

$$z^{(\text{quad})}(x,y) = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} a_{00} & a_{10} \\ a_{10} & a_{11} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_0 & b_1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + c. \quad (9)$$

On the other hand, the function $z(x, y)$ can be expanded into a Taylor series and written in terms of the abbreviations from (7) as follows:

$$\begin{aligned} z^{(\text{Taylor})}_{x_0,y_0}(x,y) &= \frac{1}{2}\begin{pmatrix} x-x_0 & y-y_0 \end{pmatrix} \\ &\quad \times \begin{pmatrix} \left.\frac{\partial^2 z}{\partial x^2}\right|_{x_0y_0} & \left.\frac{\partial^2 z}{\partial x \partial y}\right|_{x_0y_0} \\ \left.\frac{\partial^2 z}{\partial x \partial y}\right|_{x_0y_0} & \left.\frac{\partial^2 z}{\partial y^2}\right|_{x_0y_0} \end{pmatrix} \begin{pmatrix} x-x_0 \\ y-y_0 \end{pmatrix} \\ &\quad + \begin{pmatrix} \left.\frac{\partial z}{\partial x}\right|_{x_0y_0} & \left.\frac{\partial z}{\partial y}\right|_{x_0y_0} \end{pmatrix} \begin{pmatrix} x-x_0 \\ y-y_0 \end{pmatrix} + c \\ &= \frac{1}{2}\begin{pmatrix} x-x_0 & y-y_0 \end{pmatrix} \begin{pmatrix} r & s \\ s & t \end{pmatrix}\Big|_{x_0y_0} \begin{pmatrix} x-x_0 \\ y-y_0 \end{pmatrix} \\ &\quad + \begin{pmatrix} p & q \end{pmatrix}\Big|_{x_0y_0} \begin{pmatrix} x-x_0 \\ y-y_0 \end{pmatrix} + c. \end{aligned} \quad (10)$$

Apart from the factor $(1/2)$ in front of the quadratic term, this expansion has the same shape as (9). To avoid additional parameters, the notation from (7) will be directly used in the least-squares fitting, with $a_{00} = (r/2)$, $a_{11} = (t/2)$, and $a_{10} = (s/2)$. This fitting is done for each window, centered on (x_0, y_0). It is assumed that the variables x and y are defined within a local coordinate system that has the center of the window as its origin.

C. Mapping Geospatial Derivatives to Window Averages

With these assumptions, the geospatial derivatives are derived by fitting the following function to the data:

$$z^{(\text{window})}(x,y) = \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} r & s \\ s & t \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} p & q \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + c. \quad (11)$$

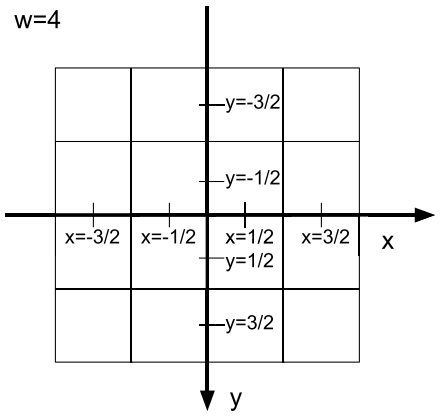

Fig. 4. Spatial coordinates within a window of size 4×4.

A least-squares fit is performed by minimizing

$$\left\langle \left(z(x,y) - z^{(\text{window})}(x,y)\right)^2 \right\rangle = \left\langle \left(z - px - qy - \frac{r}{2}x^2 - sxy - \frac{t}{2}y^2 - c\right)^2 \right\rangle \quad (12)$$

with respect to the six parameters p, q, r, s, t, and c. Toward this goal, the partial derivatives with respect to each of the parameters are set to zero. As a result, six linear equations with six unknowns are obtained. Solving this system in all generality may appear daunting, but fortunately, many of the derivatives are zero.

Fig. 4 shows the coordinate system that is used for these aggregations, for the example of a 4×4 window. In the following, only square windows of sizes that are powers of two are considered, but aggregates will still be defined and used in such a manner that there is no upper limit to the possible window sizes. The downward direction of the y-axis was chosen for practical reasons and would have to be accounted for in the computation of directional quantities that assume that zero is North or up, those for which the righthandedness of the coordinate system matters. From Fig. 4, it can be seen that the averages $\langle x \rangle$ and $\langle y \rangle$ are zero because they are taken over spatial points for which the positive and negative contributions compensate exactly. On a square grid, for each $x = x_i$, there is a point with $x = -x_i$. The term $\langle xy \rangle$ vanishes for the same reason. More generally, the sample points are chosen such that $\forall_i \exists_j (x_i = -x_j))$ and the same for y.

Of the terms that do not depend on z, only those are nonzero that have an even number of both x and y instances. All others vanish due to the symmetry. In other words, for the aggregates, which are of the shape $\langle x^n y^m z^k \rangle$, where $0 \le n \le 4$, $0 \le m \le 4$, and $0 \le k \le 1$, the averages with $k = 0$ vanish if either n or m is odd

$$\langle x^n y^m \rangle \equiv 0 \quad \text{if } (n \bmod 2 = 1) \vee (m \bmod 2 = 1). \quad (13)$$

Furthermore, the averages over x and y are independent, and $\langle x^2 y^2 \rangle = \langle x^2 \rangle \langle y^2 \rangle$, resulting in the following equations:

$$p\langle x^2 \rangle = \langle xz \rangle$$
$$q\langle y^2 \rangle = \langle yz \rangle$$

$$\begin{aligned}
\frac{r}{2}\langle x^4\rangle + \frac{t}{2}\langle x^2\rangle\langle y^2\rangle + c\langle x^2\rangle &= \langle x^2 z\rangle \\
s\langle x^2\rangle\langle y^2\rangle &= \langle xyz\rangle \\
\frac{r}{2}\langle x^2\rangle\langle y^2\rangle + \frac{t}{2}\langle y^4\rangle + c\langle x^2\rangle &= \langle y^2 z\rangle \\
\frac{r}{2}\langle x^2\rangle + \frac{t}{2}\langle y^2\rangle + c &= \langle z\rangle. \qquad (14)
\end{aligned}$$

On a square grid, points are spaced equally in the x- and y-directions, hence $\langle x^2\rangle = \langle y^2\rangle$ and $\langle x^4\rangle = \langle y^4\rangle$. Since these quantities do not depend on z, they can be calculated analytically, see the following text. Solving for p, q, r, s, t, and c gives

$$\begin{aligned}
p &= \frac{\langle xz\rangle}{\langle x^2\rangle} \\
q &= \frac{\langle yz\rangle}{\langle x^2\rangle} \\
r &= 2\frac{\langle x^2 z\rangle - \langle x^2\rangle\langle z\rangle}{\langle x^4\rangle - \langle x^2\rangle^2} \\
s &= \frac{\langle xyz\rangle}{\langle x^2\rangle^2} \\
t &= 2\frac{\langle y^2 z\rangle - \langle x^2\rangle\langle z\rangle}{\langle x^4\rangle - \langle x^2\rangle^2} \\
c &= \langle z\rangle - \langle x^2\rangle\frac{\langle x^2 z\rangle + \langle y^2 z\rangle - 2\langle x^2\rangle\langle z\rangle}{\langle x^4\rangle - \langle x^2\rangle^2}. \qquad (15)
\end{aligned}$$

Aggregates that involve z, i.e., $\langle z\rangle$, $\langle xz\rangle$, $\langle yz\rangle$, $\langle x^2 z\rangle$, $\langle y^2 z\rangle$, and $\langle xyz\rangle$, have to be computed for each window. The window aggregation will be discussed in Section II-D.

Aggregates that depend only on the spatial coordinates, including $\langle x^2\rangle$ and $\langle x^4\rangle$, remain the same for each window and can be computed analytically. For the example window with $w = 4$ in Fig. 4, the averages are computed over $x = -3/2, -1/2, 1/2$, and $3/2$, or $2\sum_{k=1}^{2}(k-1/2)^2$. Each of those terms contributes four times for each value of y. In the general case of a window of size w, each value of x contributes w times to the sum, and the total is normalized by the total number of points w^2

$$\langle x^2\rangle = \frac{1}{w^2} 2w \sum_{k=1}^{\frac{w}{2}} \left(k - \frac{1}{2}\right)^2. \qquad (16)$$

This term can be computed using power sums [61]

$$\begin{aligned}
\sum_{k=1}^{n} k &= \frac{1}{2}\left(n^2 + n\right) \\
\sum_{k=1}^{n} k^2 &= \frac{1}{6}\left(2n^3 + 3n^2 + n\right). \qquad (17)
\end{aligned}$$

Inserting (17) into (16) gives

$$\begin{aligned}
\langle x^2\rangle &= \frac{2}{w}\left(\sum_{k=1}^{\frac{w}{2}} k^2 - \sum_{k=1}^{\frac{w}{2}} k + \frac{w}{8}\right) \\
&= \frac{w^2 - 1}{12}. \qquad (18)
\end{aligned}$$

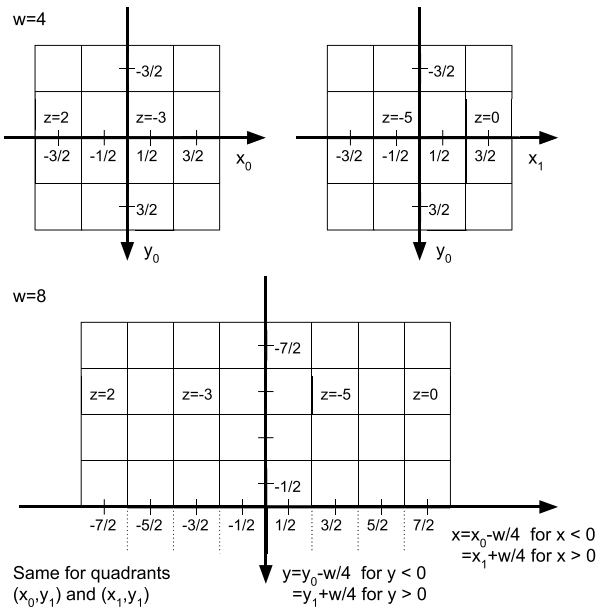

Fig. 5. Combining quadrants: (Top) two 4×4 windows are shown that are combined to form (Bottom) top two quadrants of an 8×8 window.

The fourth-order term

$$\langle x^4\rangle = \frac{1}{w^2} w \sum_{k=1}^{\frac{w}{2}} \left(k - \frac{1}{2}\right)^4 \qquad (19)$$

can be calculated similarly, using the additional power sums from [61]

$$\begin{aligned}
\sum_{k=1}^{n} k^3 &= \frac{1}{4}\left(n^4 + 2n^3 + n^2\right) \\
\sum_{k=1}^{n} k^4 &= \frac{1}{30}\left(6n^5 + 15n^4 + 10n^3 - n\right). \qquad (20)
\end{aligned}$$

The result is

$$\langle x^4\rangle = \frac{3w^4 - 10w^2 + 7}{240}. \qquad (21)$$

The frequently occurring term, $\langle x^4\rangle - \langle x^2\rangle^2$, is

$$\langle x^4\rangle - \langle x^2\rangle^2 = \frac{w^4 - 5w^2 + 4}{180}. \qquad (22)$$

This term is zero for $w = 2$, which is consistent with the observation that curvature cannot be calculated over windows of size 2 because two data points cannot be used to unambiguously fit a quadratic equation. For $w = 4$, the term is 1. For $w > 4$, the term is nontrivial, but since it can be computed algebraically, it does not contribute to the scaling of the algorithm.

D. Window Aggregates of Products of Elevation and Location

The process of simply adding smaller windows from previous iterations, which was outlined at the beginning of the section, does not work for aggregates that involve spatial coordinates because the relative coordinate systems changes. The aggregation involves shifting the coordinate system from being

centered on the windows of size $w/2$ to being centered on the combined window of size w. Fig. 5 specifically shows the changes in the x-coordinate of the coordinate system. When computing aggregates over 4×4 windows, the coordinate systems are centered on those windows. In the next iteration, which uses 8×8 windows, the coordinate system of each 4×4 quadrant has to be shifted accordingly. The same logic is applied to the y-coordinate.

The z-values for the full window are given based on the z-values of the constituent quadrants as follows, where the notation z_{00} is used for $z_{i_0 j_0}$, z_{10} for $z_{i_0+w/2, j_0}$, and so on:

$$z(x,y) = \begin{cases} z_{00}(x_0, y_0), & \text{for } x < 0, y < 0 \\ z_{10}(x_1, y_0), & \text{for } x > 0, y < 0 \\ z_{01}(x_0, y_1), & \text{for } x < 0, y > 0 \\ z_{11}(x_1, y_1), & \text{for } x > 0, y > 0 \end{cases} \tag{23}$$

where $x_0 = x + (w/4)$, $x_1 = x - (w/4)$, $y_0 = y + (w/4)$, and $y_1 = y - (w/4)$. Notice that when averages over quadrants were computed in the previous iteration, there was only one definition of x, and that definition involved a different coordinate system for each of the windows that are now aggregated. In the following, those four coordinate systems have to be distinguished such that a systematic coordinate transformation can be done into the shared coordinate system of the new iteration. Values for each of the coordinates within quadrants are within the range $-(w/4) < x_0, x_1, y_0, y_1 < (w/4)$.

For the averaged value of z, the shifting of coordinates does not have an impact on the calculation of averages, and the averaged value $\langle z_{00} \rangle$ is simply the sum of the averages that were computed in the previous iteration $\langle z \rangle_{00}$

$$\begin{aligned} \langle z \rangle &= \frac{1}{4}\left(\langle z_{00} \rangle + \langle z_{10} \rangle + \langle z_{01} \rangle + \langle z_{11} \rangle\right) \\ &= \frac{1}{4}\left(\langle z \rangle_{00} + \langle z \rangle_{10} + \langle z \rangle_{01} + \langle z \rangle_{11}\right) \\ &= \langle z \rangle_{\left[\substack{++\\++}\right]}. \end{aligned} \tag{24}$$

The last line introduces a simplified notation that indicates that, for this average, the quadrants are all added. For other averages, some quadrants may be subtracted, so a notation is used that specifies the sign used in the aggregation of each of the quadrants

$$\langle z \rangle_{\left[\substack{++\\++}\right]} = \langle z \rangle_{\begin{bmatrix} \text{top-left quadr.} & \text{top-right quadr.} \\ \text{bottom-left quadr.} & \text{bottom-right quadr.} \end{bmatrix}}. \tag{25}$$

For averages over products of z and x or y, the coordinate shifts affect the values of the average because the shifting itself contributes an additional term

$$\begin{aligned} \langle xz \rangle = \frac{1}{4}\Big(&\Big\langle \Big(x_0 - \frac{w}{4}\Big) z \Big\rangle_{00} + \Big\langle \Big(x_1 + \frac{w}{4}\Big) z \Big\rangle_{10} \\ &+ \Big\langle \Big(x_0 - \frac{w}{4}\Big) z \Big\rangle_{01} + \Big\langle \Big(x_1 + \frac{w}{4}\Big) z \Big\rangle_{11} \Big). \end{aligned} \tag{26}$$

The goal is, as before, to write the expression as a function of the averages that were computed in the previous iteration, during which the window size was $(w/2)$. When doing so, $\langle x_0 z_{01} \rangle$ is replaced with $\langle xz \rangle_{01}$ and so on because x_0 equals x of the previous iteration, and the subscript to the average indicates that it is a previous iteration average. Constant factors are also pulled out of the averages, e.g., $\langle (w/4) z \rangle_{01} = (w/4)\langle z \rangle_{01}$. Averages over sums are broken up into the sums of averages

$$\begin{aligned} \langle xz \rangle = \frac{1}{4}\Big(&\langle xz \rangle_{00} + \langle xz \rangle_{10} + \langle xz \rangle_{01} + \langle xz \rangle_{11} \\ &+ \frac{w}{4}\left(\langle z \rangle_{10} + \langle z \rangle_{11} - \langle z \rangle_{00} - \langle z \rangle_{01}\right)\Big). \end{aligned} \tag{27}$$

Some of the quadrant averages of z now appear with a negative sign. The notation that was introduced in (25) will now be used to keep track of the signs

$$\langle xz \rangle = \langle xz \rangle_{\left[\substack{++\\++}\right]} + \frac{w}{4}\langle z \rangle_{\left[\substack{-+\\-+}\right]}. \tag{28}$$

The linear term $\langle xz \rangle_{\left[\substack{++\\++}\right]}$ is the sum of all four terms linear in x from the previous iteration. The constant term $\langle z \rangle_{\left[\substack{-+\\-+}\right]}$ is the difference between the averages of the constant terms with positive x ($\langle z \rangle_{10}$ and $\langle z \rangle_{11}$) and with negative x ($\langle z \rangle_{00}$ and $\langle z \rangle_{01}$). Equivalent statements hold for $\langle yz \rangle$, and the difference is correspondingly taken between values with positive and negative y values

$$\begin{aligned} \langle yz \rangle &= \frac{1}{4}\Big(\langle yz \rangle_{00} + \langle yz \rangle_{10} + \langle yz \rangle_{01} + \langle yz \rangle_{11} \\ &\quad + \frac{w}{4}\left(\langle z \rangle_{01} + \langle z \rangle_{11} - \langle z \rangle_{00} - \langle z \rangle_{10}\right)\Big) \\ &= \langle yz \rangle_{\left[\substack{++\\++}\right]} + \frac{w}{4}\langle z \rangle_{\left[\substack{--\\++}\right]}. \end{aligned} \tag{29}$$

Averages of terms that are quadratic in the coordinates are evaluated similarly

$$\begin{aligned} \langle x^2 z \rangle &= \frac{1}{4}\Big(\Big\langle \Big(x_0 - \frac{w}{4}\Big)^2 z \Big\rangle_{00} + \Big\langle \Big(x_1 + \frac{w}{4}\Big)^2 z \Big\rangle_{10} \\ &\quad + \Big\langle \Big(x_0 - \frac{w}{4}\Big)^2 z \Big\rangle_{01} + \Big\langle \Big(x_1 + \frac{w}{4}\Big)^2 z \Big\rangle_{11} \Big) \\ &= \langle x^2 z \rangle_{\left[\substack{++\\++}\right]} + \frac{w}{2}\langle xz \rangle_{\left[\substack{-+\\-+}\right]} + \frac{w^2}{16}\langle z \rangle_{\left[\substack{++\\++}\right]} \end{aligned} \tag{30}$$

$$\langle y^2 z \rangle = \langle y^2 z \rangle_{\left[\substack{++\\++}\right]} + \frac{w}{2}\langle yz \rangle_{\left[\substack{--\\++}\right]} + \frac{w^2}{16}\langle z \rangle_{\left[\substack{++\\++}\right]}. \tag{31}$$

Notice that these quadratic terms involve differences between quadrants of linear terms $\langle xz \rangle_{\left[\substack{-+\\-+}\right]}$, much as the linear terms involved differences between quadrants of constant terms $\langle z \rangle_{\left[\substack{-+\\-+}\right]}$. However, these terms do not have to be stored beyond the iteration in which they are used to compute the fit parameters. Only the complete window averages $\langle z \rangle$, $\langle xz \rangle$, $\langle yz \rangle$, $\langle x^2 z \rangle$, $\langle y^2 z \rangle$, and $\langle xyz \rangle$ are used in the next iteration.

The term $\langle xyz \rangle$ furthermore includes differences of quadrant-level $\langle xz \rangle$ averages in the y-direction and corresponding $\langle yz \rangle$ differences in the x-direction, as well as a term in which off-diagonal $\langle z \rangle$ aggregates are subtracted from diagonal ones

$$\begin{aligned} \langle xyz \rangle = \frac{1}{4}\Big(&\Big\langle \Big(x_0 - \frac{w}{4}\Big)\Big(y_0 - \frac{w}{4}\Big) z \Big\rangle_{00} \\ &+ \Big\langle \Big(x_1 + \frac{w}{4}\Big)\Big(y_0 - \frac{w}{4}\Big) z \Big\rangle_{10} \\ &+ \Big\langle \Big(x_0 - \frac{w}{4}\Big)\Big(y_1 + \frac{w}{4}\Big) z \Big\rangle_{01} \end{aligned}$$

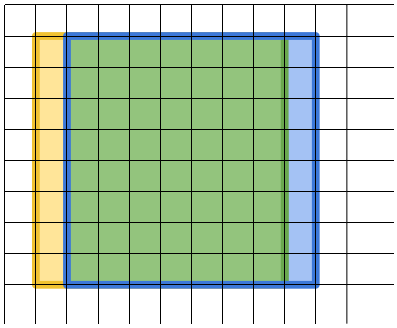

Fig. 6. Schematic showing the cancellation of terms when subtracting neighboring 8×8 pixel windows.

$$
\begin{aligned}
&+\left\langle \left(x_1+\frac{w}{4}\right)\left(y_1+\frac{w}{4}\right)z \right\rangle_{11}\Bigg) \\
&= \langle xyz \rangle_{\left[\substack{++\\++}\right]} + \frac{w}{4}\langle xz \rangle_{\left[\substack{--\\++}\right]} \\
&\quad + \frac{w}{4}\langle yz \rangle_{\left[\substack{-+\\-+}\right]} + \frac{w^2}{16}\langle z \rangle_{\left[\substack{+-\\-+}\right]}. \qquad (32)
\end{aligned}
$$

$\langle xz \rangle_{\left[\substack{--\\++}\right]}$ can be viewed as the difference between the linear dependence of z on x in the quadrants with positive y and the quadrants with negative y. $\langle z \rangle_{\left[\substack{+-\\-+}\right]}$ is a combination of constant terms for which the sign is that of the product of the signs of x and y. This concludes the derivation of equations that were outlined in Fig. 3.

E. Comparison With Two-Step Process

It may be tempting to think that computing curvature over a smoothed surface alone should itself resolve problems of noise adequately. Unfortunately, a two-step process has important limitations. Consider the schematic in Fig. 6, which shows two 8×8 windows that are one raster point apart, with the left window shown in yellow and the right one in blue. The area that is covered by both windows is shown in green.

If the values in the yellow window are subtracted from those in the blue window, only the narrow yellow and the blue strips on the left and right side contribute to the difference. The values in the green region cancel and do not contribute to the difference at all. This also means that if the data are noisy, the averaging over the noise only happens within the narrow yellow and blue strips.

This schematic oversimplifies the difference-taking to involve no more than 2 pixels. In the actual two-step approach of using smoothing followed by a 3×3 window curvature computation, there would be two rows of raster points on each side contributing. However, the general problem remains that points other than those at the periphery cancel and do not contribute to the overall curvature computation. The smoothing that worked well for the landscape itself is partially undone in the curvature computation.

That problem does not apply to the proposed window-aggregation algorithm, which averages gradients over all relevant points and is equivalent to a standard quadratic fit over the window in question. As such, the averaging of errors extends to all points in the window. The difference will be visible in the evaluation, where it will be seen that the window-aggregation algorithm is less prone to noise than the conventional two-step analysis.

F. Analytically Computed Ground Truth

For remotely sensed DEM data, there is no "correct" slope or curvature available that could serve as ground truth. A dataset was therefore constructed that allows for an analytical evaluation of curvature measures while offering some diversity in observable features. As such, the model had to be twice differentiable throughout the region, and the formulas for all derivatives had to be encoded in the implementation. This requirement made it highly impractical to use functions that are defined over segments.

A common model for a spatial structure that is differentiable arbitrarily many times and can be represented in arbitrarily many dimensions, while being bounded and decreasing toward the edges of any region that contains it, is the Gaussian function $e^{-(x^2+y^2)/\sigma^2}$. Its convenient properties make it a go-to function in most physical disciplines, including geosciences. However, in this basic form, the model only allows for a very limited set of curvature combinations. However, the convenient properties of the Gaussian function apply even in superposition. In other words, all necessary derivatives could be encoded in a parameterized fashion, and analytical derivatives computed, even when the individual hills themselves may no longer be distinguishable to the observer.

The resulting model still has the problem that it does not account for noisy and nondifferentiable scenarios in the real world. When adding a noise model, it had to be established that the analytic derivatives could be expected to continue holding. For this reason, the simplest noise model was used of individually adding random numbers that follow a normal distribution. With this noise model, derivatives are no longer strictly the same as in the noise-free model, but it can be expected that the noise cancels increasingly well for increasing window sizes in any form of window-based smoothing.

Notice that the slope and curvature values represent point-based slopes and curvatures, while the proposed algorithm returns slopes and curvatures overextended and potentially large windows. Differences between both are expected and can indeed be seen in the evaluation. One could consider applying window-based integration to the ground truth, but attempting to do so would raise more questions than it would answer. One question is whether second derivatives or geometric curvatures should be integrated. Both definitions would raise concerns, either related to the computation or to the role as ground truth.

Instead, window-based slopes and curvatures are viewed as approximates to the point-based slopes and curvatures for the specific context of the artificially generated landscapes. In this viewpoint, the window-based computation serves the purpose of eliminating high-frequency noise, and any change to the slope and curvature determination for the landscape is viewed as error. This "error" increases with increasing window size, as shown in Fig. 14 for a window size of $w = 32$. What may be most surprising is that, despite this expected difference, and

for large enough noise, increasing window sizes can still result in decreasing error. These findings will be discussed in more detail in Section IV-A.

The definition of the artificial landscape is given as follows:

$$f(x,y) = \sum_{i=0}^{K} e^{-\frac{(x-\mu_i)^2+(y-\nu_i)^2}{2\sigma_i^2}} \tag{33}$$

where K is the number of peaks, σ_i is the width of peak i, and μ_i and ν_i are the coordinates of the center of peak i. In the experiments, the number of peaks, K, was 10, σ varied from 10% to 30% of the image size, and μ and ν were randomly selected within the image.

The partial derivatives of this function can be taken analytically. Since the window size is not taken into consideration, it is to be expected that the analytical values would be poor approximations when the window size is comparable to σ. In the evaluation, the largest window size was 64, and σ was chosen to be as small as 10% of the image size of 512, so some level of inaccuracy is expected. The partial derivatives of the above artificial landscape are given as follows:

$$\begin{aligned}
\frac{\partial f(x,y)}{\partial x} &= -\sum_{i=0}^{K} \frac{x-\mu_i}{\sigma_i^2} e^{-\frac{(x-\mu_i)^2+(y-\nu_i)^2}{2\sigma_i^2}} \\
\frac{\partial f(x,y)}{\partial y} &= -\sum_{i=0}^{K} \frac{y-\nu_i}{\sigma_i^2} e^{-\frac{(x-\mu_i)^2+(y-\nu_i)^2}{2\sigma_i^2}} \\
\frac{\partial^2 f(x,y)}{\partial x^2} &= \sum_{i=0}^{K} \frac{(x-\mu_i)^2-\sigma_i^2}{\sigma_i^4} e^{-\frac{(x-\mu_i)^2+(y-\nu_i)^2}{2\sigma_i^2}} \\
\frac{\partial^2 f(x,y)}{\partial y^2} &= \sum_{i=0}^{K} \frac{(y-\nu_i)^2-\sigma_i^2}{\sigma_i^4} e^{-\frac{(x-\mu_i)^2+(y-\nu_i)^2}{2\sigma_i^2}} \\
\frac{\partial^2 f(x,y)}{\partial x \partial y} &= \sum_{i=0}^{K} \frac{(x-\mu_i)(y-\nu_i)}{\sigma_i^4} e^{-\frac{(x-\mu_i)^2+(y-\nu_i)^2}{2\sigma_i^2}}.
\end{aligned} \tag{34}$$

The topographic functions are computed from the derivatives as given in (7).

III. Implementation

A. Array-Based Implementation

The implementation was done in Python using the array-based processing capabilities of NumPy. It is available on Github at https://github.com/amdenton/SlidingWindows

For performing the aggregation, four copies of the arrays that hold the raster image, corresponding to each of the quadrants, are "flattened", i.e., treated as a 1-D array. The four quadrants are treated as columns in a temporary array. Each of the columns represents a shifted version of the aggregate raster. Aggregation is then performed in the row direction of the temporary array. Aggregates that correspond to points outside the frame of $(M-w+1) \times (N-w+1)$ meaningful windows are deleted.

Fig. 7 shows the concept. The input raster contains results from an aggregation of 2×2 windows and is shown on the left-hand side. Four copies of the flattened version of this raster are shown on the right-hand side, each with a different offset. The first one has an offset of 0, corresponding to the top-left raster point. The second has an offset of $\delta = 2$, representing the values from the top-right quadrant. The aggregate value representing the bottom-left quadrant is stored $\delta = 2$ rows below the first raster point. In the flattened raster, this corresponds to a shift of $M * \delta$, where M is the overall width of the DEM. For the tiny example of an image that is only five raster points wide, this shift is 10, and hence, the third copy of the flattened input raster is shown with an offset of 10. The final copy corresponds to the bottom-right quadrant and has a shift of $M * \delta + \delta = 12$.

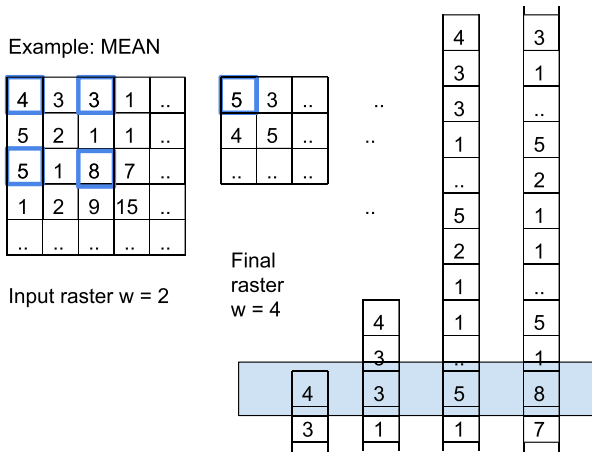

Fig. 7. Schematic of the array-based implementation of the raster aggregation step from $w = 2$ to $w = 4$. The example corresponds to an image size of $N = 6$ and $M = 6$, which was reduced to a 5×5 raster at $w = 2$ and is further reduced to a 3×3 output at window size $w = 4$.

It would be straightforward to implement the approach for use on a graphical processing unit (GPU) and easier than implementing multipoint algorithms such as Wu *et al.* [62] did. From a processing perspective, the proposed algorithm makes heavy use of basic operations on large arrays and can therefore be optimized relatively easily.

B. IART Algorithm

The pseudocode for the proposed IART algorithm is depicted in Algorithm 1. In addition to the basic structure that was shown in Fig. 3, more detail was added to flesh out steps that were derived in Section II. The algorithm has two main parts. The aggregation step in procedure AGGREGATE progressively aggregates the rasters representing $\langle z \rangle$, $\langle xz \rangle$, $\langle yz \rangle$, $\langle x^2 z \rangle$, $\langle y^2 z \rangle$, and $\langle xyz \rangle$, in each step doubling the window size. The procedure CURVATURES uses the aggregates to compute curvatures.

The $\langle z \rangle$ raster is initialized to the original DEM in procedure INITIALIZE. The initialization corresponds to a window size of $w = 1$. At that window size, all rasters correspond to an average over a single value with $x = 0$ and $y = 0$, within the coordinate system that is centered on that value. Hence, all aggregates other than $\langle z \rangle$ are initialized to 0, implying that the entire representation of spatial coordinates stems from the shift in the coordinate systems upon aggregation of the four quadrants from the previous iteration. In the first aggregation step, which has an output window size of $w = 2$, the shift

Algorithm 1 Curvature Using Iterative Aggregation

procedure INITIALIZE
  $aggs \leftarrow 0$
  $z \leftarrow import\_dem()$
  $xz, yz, xxz, yyz, xyz \leftarrow zeros(\text{size}(z))$
return $aggs, z, xz, yz, xxz, yyz, xyz$
procedure AGGREGATE($aggs, z, xz, yz, xxz, yyz, xyz$)
  $w = 2^{aggs}$
  $z_{new} \leftarrow add\_all(z)$ ▷ See eqn. (24)
  $xz_{new} \leftarrow add\_all(xz)$
  $+w/4 * add\_right(z)$ ▷ See eqn. (28)
  $yz_{new} \leftarrow add\_all(yz)$
  $+w/4 * add\_bottom(z)$ ▷ See eqn. (29)
  $xxz_{new} \leftarrow add\_all(xxz) + w/2 * add\_right(xz)$
  $+w^2/16 * add\_all(z)$ ▷ See eqn. (30)
  $yyz_{new} \leftarrow add\_all(yyz) + w/2 * add\_bottom(yz)$
  $+w^2/16 * add\_all(z)$ ▷ See eqn. (31)
  $xyz_{new} \leftarrow add\_all(xyz)$
  $+w/4 * add\_bottom(xz) + w/4 * add\_right(yz)$
  $+w^2/16 * add\_diag(z)$ ▷ See eqn. (32)
  $aggs = aggs + 1$
return $aggs, z_{new}, xz_{new}, yz_{new}, xxz_{new}, yyz_{new}, xyz_{new}$
procedure CURVATURES($aggs, z, xz, yz, xxz, yyz, xyz$)
  $w = 2^{aggs}$
  $xx = (w^2 - 1)/12$ ▷ See eqn. (18)
  $xmult = (w^4 - 5 * w^2 + 4)/180$ ▷ See eqn. (22)
  $p = xz/xx$
  $q = yz/xx$
  $r = 2 * (xxz - (xx * z))/xmult$
  $s = xyz/(xx * xx)$
  $t = 2 * (yyz - (xx * z))/xmult$
  ▷ For the derivation of $p, q, r, s,$ and t see eqns. (15)
  $slope = arctan(\sqrt{p^2 + q^2})$
  $d0 = (p^2 + q^2) * \sqrt{1 * p^2 + q^2}^3$
  $profile = -(p^2 * r + 2 * p * q * s + q^2 * t)/d0$
  $d1 = (p^2 + q^2) * \sqrt{1 * p^2 + q^2}$
  $tangential = -(q^2 * r - 2 * p * q * s + p^2 * t)/d1$
  ▷ From literature, see eqns. (7)
return *slope, profile, tangential*
procedure MAIN(*file_name,max_aggs*)
  ▷ Input: DEM file name, max number of aggregations
  $aggs, z, xz, yz, xxz, yyz, xyz \leftarrow$ INITIALIZE()
  while $aggs <= max\_aggs$ **do**
    $aggs, z, xz, yz, xxz, yyz, xyz \leftarrow$
    AGGREGATE($aggs, z, xz, yz, xxz, yyz, xyz$)
    if $aggs > 1$ **then**
      ▷ Curvature requires at least 2 aggregations
      slope, profile, tangential $\leftarrow$
      CURVATURES($aggs, z, xz, yz, xxz, yyz, xyz$)
      Output(*aggs, slope, profile, tangential*)

covers a distance of $(w/4) = (1/2)$. Later aggregations shift the coordinate systems by integer values that are powers of 2.

In this pseudocode, curvatures are computed for each aggregation step, i.e., profile and tangential curvature rasters are output for all window sizes that are powers of 2 up to the largest one. If only the curvatures for the largest window size were of interest, the procedure CURVATURES would only have to be called once at the end. As presented, all intermediate curvatures are returned, as is appropriate when a multiscalar analysis is of interest. For simplicity, the pseudocode also includes the slope computation among the curvatures rather than as a faster limited-purpose slope function.

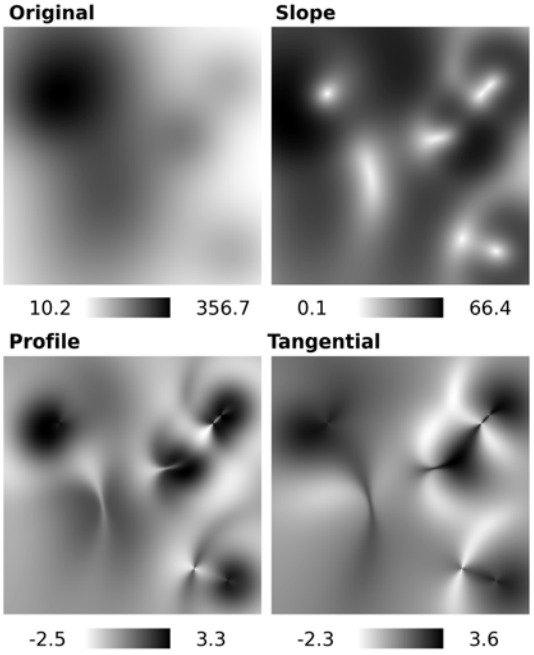

Fig. 8. (Top Left) Artificial dataset consisting of ten Gaussian hills of height 100. (Top Right) Slope of that dataset. (Bottom Left) Profile curvature. (Bottom Right) Tangential curvature. The sign conventions are given as in [13].

IV. EXPERIMENTS

A. Evaluation on Artificial Data

As a first test, the IART algorithm is applied to an artificially created landscape as described in Section II-F. Specifically, ten Gaussian hills were created with randomly selected centers and widths chosen randomly within a range of 10%–30% of the image size. The height is chosen to be 100 m for each hill, assuming a raster size of 1 m. The intention of using such a large height is to detect problems more effectively. For the sake of reproducibility, the same seed (1) was used for the entire evaluation. Table II shows the values for the centers (μ and ν) and widths of the Gaussian hills (σ) as a fraction of the image size.

Fig. 8 shows the resulting image in the top-left corner. Some of the peaks can be seen individually, and some merge and create ridges, such as one toward the left of the image. Moreover, troughs can be seen around some of the hills, such as for a hill in the bottom-right corner. The top-right panel shows the slope of the landscape, as calculated by (34). The tops of the Gaussian hills have zero slope, as expected. There is also a ridge toward the left that shows a negligible slope.

TABLE II
CENTERS AND WIDTH OF GAUSSIAN HILLS USED AS ARTIFICIAL DATASET

Index	μ	ν	σ
0	0.72	0.42	0.10
1	0.15	0.30	0.12
2	0.35	0.19	0.18
3	0.42	0.54	0.24
4	0.88	0.20	0.11
5	0.42	0.67	0.21
6	0.20	0.14	0.26
7	0.31	0.97	0.24
8	0.89	0.88	0.12
9	0.17	0.04	0.28

The largest slopes can be seen near a tall hill in the top-left corner, which is a superposition of primarily the four out of ten hills that are located in the top-left quadrant in the landscape.

The bottom-left panel shows the profile curvature. It is largest for steep hills with small σ, except where those hills show ridges in the direction of other hills. The ridge toward the left shows small profile curvature specifically on the top portion, where the slope is directed along the ridge. The tangential curvature in the bottom-right panel, in contrast, is large around those ridges and smallest in the troughs around hills that are at the base level of the landscape and hence largely constant perpendicular to the slope. Note that all curvatures are positive when downward, see (8), following the mathematical definition that is used in the article by Florinsky [13], but not of the ArcGIS software, which uses a positive sign for upward profile curvature.

When processing this image with a sliding window of increasing sizes up to 64 $\times$ 64, without added noise, with either the proposed IART algorithm or ArcGIS, there were no obvious changes even for the largest window sizes. This is to be expected since the Gaussian nature of the landscape makes it relatively smooth in comparison with what one may find in a real landscape. The widths of the Gaussian hills was selected such that the window size would rarely exceed σ.

The analytically calculated topographic variables were then compared with their numerically computed counterparts and evaluated using the MAPE, as shown in Fig. 9. For the evaluation using ArcGIS, the sign was adjusted to match the theoretical definitions, which meant reversing the sign of the profile curvature. For window sizes of $w > 3$, the following processing was used. Focal statistics was used with a square window of the size $w_f = w - 2$ followed by the conventional curvature computations using a 3 $\times$ 3 window. Since both processing types amount to a convolution, this approach results, in total, in effective windows of $w \times w$ raster points. Since the periphery of the resulting image were highly distorted, a $w/2$ frame was removed such that the comparison with the IART algorithm, for which the frame is removed as part of the process, would be fair.

For slope, there is relatively little difference between the theoretical and actual values (the top of Fig. 9) until the largest window size of 64 $\times$ 64. Since the $w = 64$ is greater than some of the σ values of hills, which can be as small as 10% of the 512 image size or 51 raster points, substantial differences

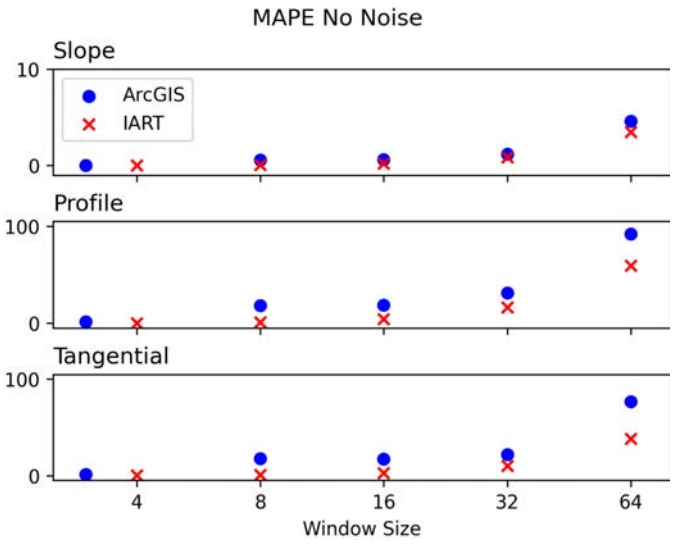

Fig. 9. MAPE in comparison with theoretical values for the artificial data with no noise, for the IART algorithm (red crosses), and ArcGIS (blue circles). Variables of interest: (Top row) slope, (second row) profile curvature, and (Bottom row) tangential curvature. For ArcGIS the processing includes smoothing (for window sizes >=8) followed by (Top row) slope, (second row) profile, and (Bottom row) planform curvature computation.

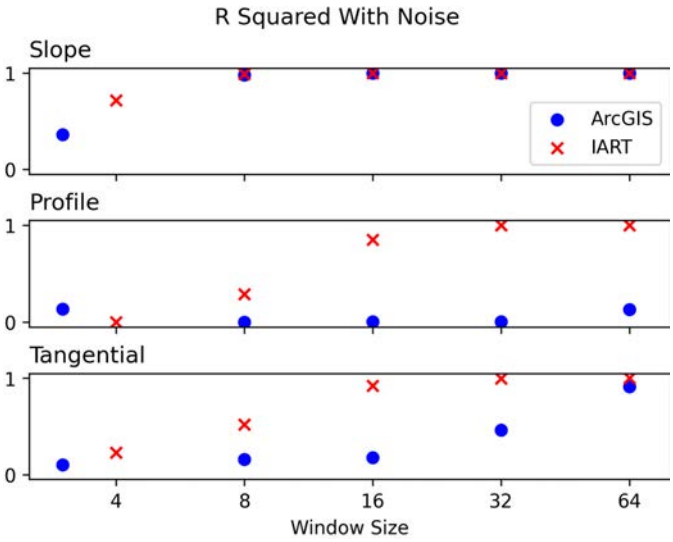

Fig. 10. R^2 in comparison with theoretical values for the artificial data with noise for the IART algorithm (red crosses) and ArcGIS (blue circles). Variables of interest: (Top row) slope, (second row) profile curvature, and (Bottom row) tangential curvature. For ArcGIS the processing includes smoothing (for window sizes >=8) followed by (Top row) slope, (second row) profile, and (Bottom row) planform curvature computation.

are expected and can be seen. For profile curvature (second panel), the differences in comparison with the theoretical values are higher, but still only major at the largest window size.

The comparisons so far do not provide evidence for benefits of using large window size since, in all cases, accuracy declined with increasing window size. The situation changes when random noise of up to 10% of the maximum height of the hill is added. In fact, when doing so, the MAPE became so

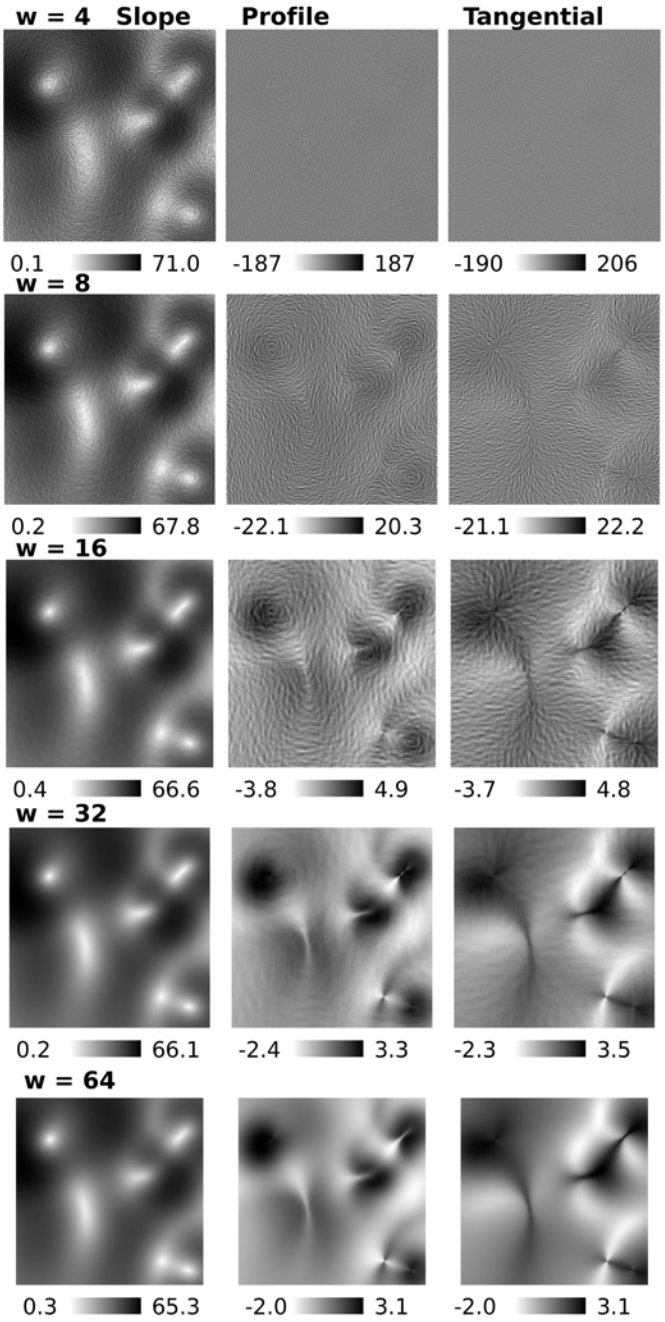

Fig. 11. (Left) Slope, (Middle) profile curvature, and (right) tangential curvature using the IART algorithm on artificial data with noise, for window sizes from top to bottom: $w = 4$, 8, 16, 32, and 64.

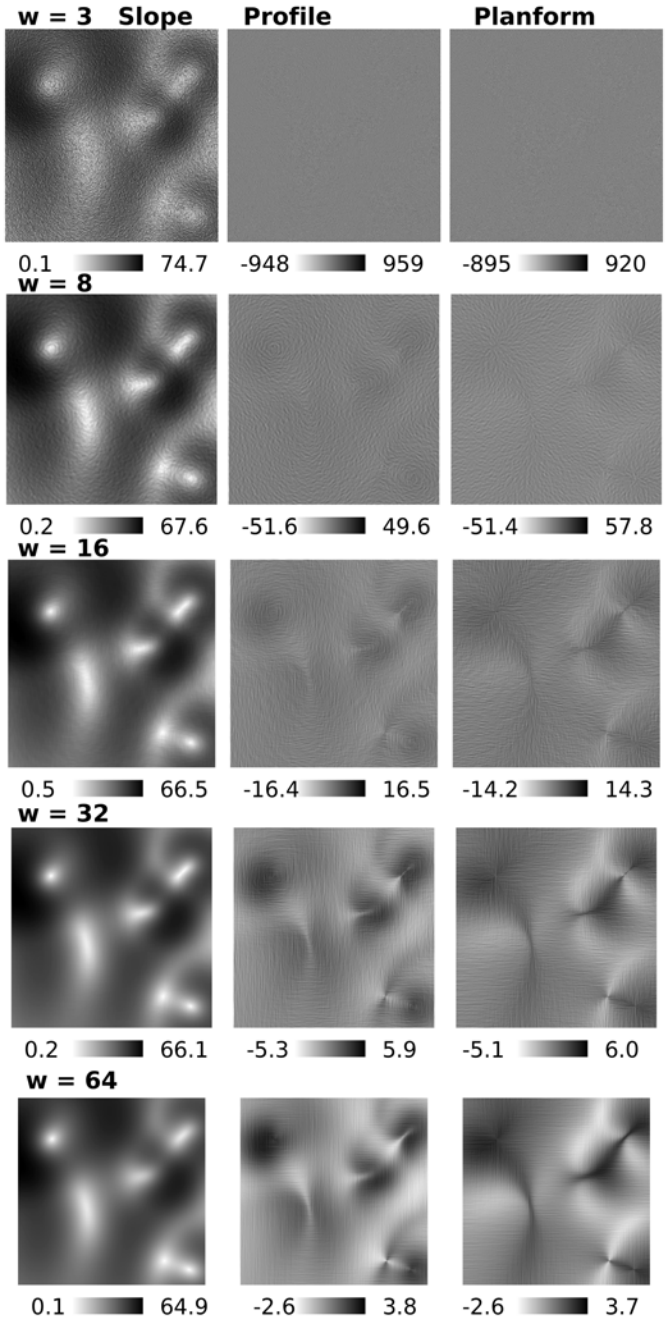

Fig. 12. (Left) Slope, (Middle) profile curvature, and (Right) planform curvature using ArcGIS on artificial data with noise for effective window sizes from top to bottom 3 (no smoothing), 8, 16, 32, and 64, i.e., smoothing using focal statistics with window sizes 6, 14, 30, and 62, respectively (see the text).

large throughout the experiment that it was no longer a useful measure of estimation quality.

Instead, the R^2 value was used, as can be seen in Fig. 10. Note that, in contrast to MAPE, a large value is now desirable. R^2 values for all algorithms suggest that the results are largely useless for the smallest window sizes, which are 3×3 in the case of ArcGIS and 4×4 in the case of the IART algorithm. Because the noise was added at the level of individual raster points, the topographic features that can be attributed to the noise overwhelm the underlying artificial landscape. For larger window sizes, slope has good R^2 values for both ArcGIS and the IART algorithm, but curvature is still problematic. Curvature remains problematic for both ArcGIS and the IART algorithm for a window size of 8×8.

When the curvature is computed over windows of size 16×16 and larger, the IART algorithm has reasonably high R^2 values even in this noisy setting. Yet, ArcGIS curvatures are still not very useful, despite the substantial smoothing of larger window sizes. For the largest window sizes (32×32 and 64×64), R^2 for the IART algorithm is very close to 1. This

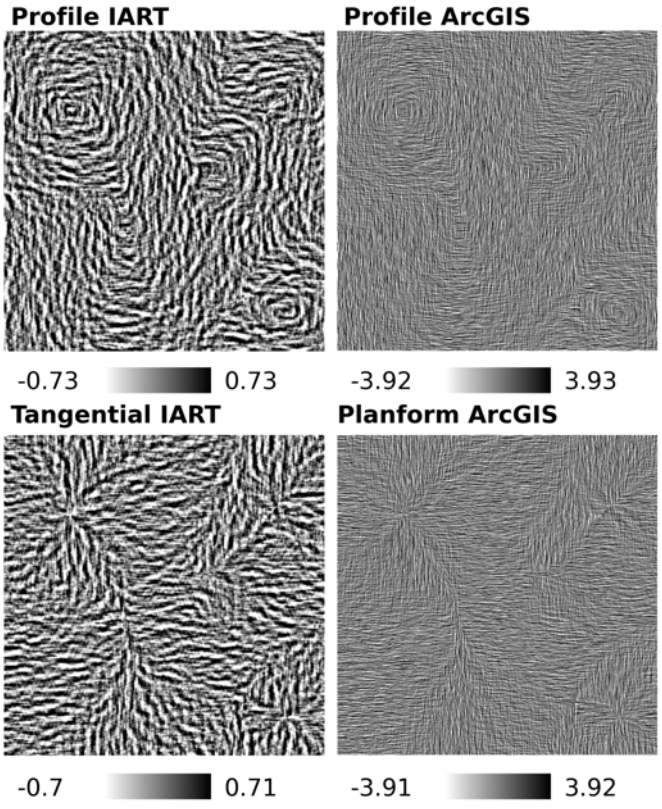

Fig. 13. Difference between numerical and theoretical curvatures for $w = 16$ using artificial data with noise. The output is normalized to the range between the 10th and 90th percentiles of occurring values.

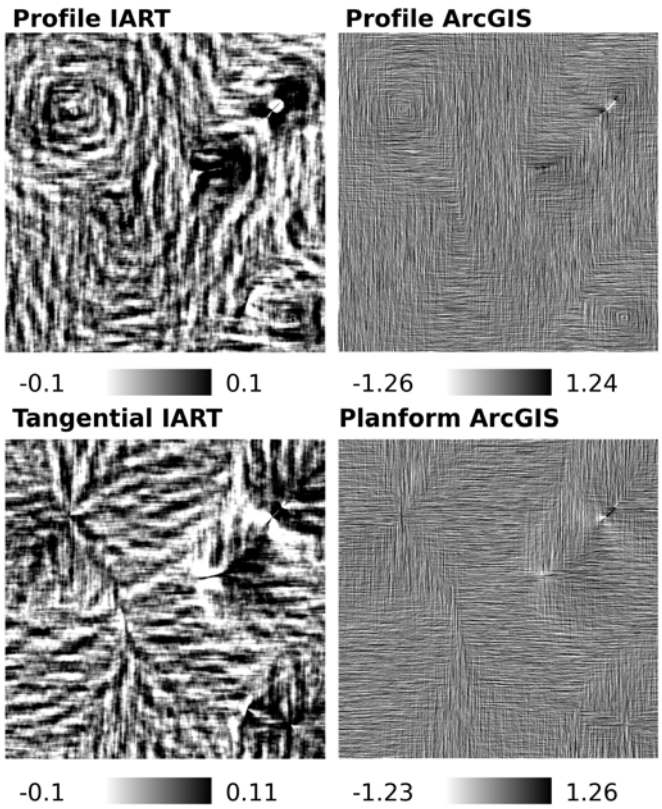

Fig. 14. Difference between numerical and theoretical curvatures for $w = 32$ using artificial data with noise. The output is normalized to the range between the 10th and 90th percentiles of occurring values.

is the case although the theoretical slope and curvatures do not account for the window size at all. In other words, the benefits due to the averaging over noisy data points are overwhelmingly more important than any drawbacks due to the theoretically poorer match. This is the case even for ArcGIS, although the ArcGIS results do not approach $R^2 = 1$ until the largest window size of 64×64 and are lower than the results of the IART algorithm even then.

The differences in quality between the IART output and ArcGIS's curvature results can be seen in Figs. 11 and 12. The figures show slope (left), profile (middle), and plan (right) for increased window sizes from top to bottom. While slope is acceptable for all but the unsmoothed image, curvatures do not become clear until the lower rows of the images. In Fig. 11, it can be seen that the IART algorithm produces the characteristic features with clarity for window sizes $w = 32$ and $w = 64$, and even the results for a 16×16 window provide the detailed output albeit with some noise. The ArcGIS output is much noisier in comparison.

Note that the images are normalized to the maximum and the minimum of the range of occurring numbers. The consistently much larger ranges of the ArcGIS values are indicators of the much more substantial noise in those images. Even when it may be possible to detect the features of the underlying image visually, any computational downstream analysis would suffer from that noise. For example, for window size $w = 32$, the curvatures can be discerned even in the ArcGIS output, but the displayed range of values is more than twice that of the IART output. Those large random values would negatively affect any further processing.

Fig. 13 shows the differences between window-based and ground-truth results at the level of individual raster points, for a window size of $w = 16$. This visualization allows analyzing the differences between the numerical and theoretical curvatures in more detail. It can be seen that the noise is dominating both the result of the proposed IART algorithm and the ArcGIS comparison approach. However, the range of the IART output is smaller by about a factor of 5. The ArcGIS output shows such rapid fluctuations that they can be barely distinguished. Note that the fluctuations are largely horizontal and vertical, which matches the theoretical considerations in Section II-E. When applying curvature computations on smoothed images, curvature is effectively computed as differences between one- or two-pixel columns and rows. The horizontal and vertical streaks in the output are tell-tale signs that noise fluctuations are averaged in a predominantly vertical or horizontal fashion, but not in a way that uses the entire window.

One might consider increasing the window size yet further to achieve more smoothing, but doing so creates inaccuracies that cannot be prevented because a larger region in the image is summarized with each window. To test the effect of inaccuracies due to large window sizes, the analysis from Fig. 13 was repeated for a window size of 32×32. The

Fig. 15. 500 × 500 DEM showing a tributary to the Souris River in Ward County North Dakota.

output can be seen in Fig. 14. First, it can be noted that for both algorithms, the errors are reduced, although they are more substantially reduced for the IART algorithm than for ArcGIS, and the ArcGIS output at window size 32 × 32 is noisier than the IART output for window size 16 × 16. Second, while the random noise is still substantial for ArcGIS, the size of the smoothing window is already creating distortions that are noticeable as errors. Distortions are a mathematically quantifiable consequence of window size and cannot be avoided in either approach, but when the noise reduction goals of window smoothing have effectively succeeded at the $\omega = 32$ window size for IART, the ArcGIS errors are still very large. Notice that the scale of the ArcGIS errors is a full order of magnitude larger than that of IART errors. As such, the IART algorithm is far more effective at accomplishing the goals of window smoothing toward noise reduction.

B. Evaluation on DEM Data

The evaluation on real elevation data uses a DEM of size 500 × 500 with 1-m resolution from a tributary to the Souris River in Ward County, North Dakota. These data come from QA Program for High Resolution Lidar Data for Multiple Counties in North Dakota, Phase 7, and specifically from the Ward County block [63]. The data were collected in 2017 and published in February 2018. It has 1-m spatial resolution and was collected to have 0.1-m vertical accuracy. The DEM is in NAD83_UTM_zone_14N projection (easting 314 800 and northing 5 359 750) The DEM itself can be seen in Fig. 15.

Fig. 16 shows the result of processing with the IART algorithm, following the same steps as in Section IV-A, except that the ground truth for this landscape is unknown. As previously, the results in the top row, which correspond to a $\omega = 4$ window size, show a substantial level of noise. This noise can also be observed quantitatively from the large range in which both profile and tangential curvature are observed. For increasing window sizes, details become increasingly clearer and the range of curvatures decreases. At window size $\omega = 16$, in the third row, substantial detail about the water systems in this area can be identified. The profile image provides information along the flow direction into the water systems, while the tangential image elucidates many additional tiny ridges and creeks perpendicular to the slope, which, respectively, disperse or condense the flows. The next larger window sizes of $\omega = 32$ removes yet more noise while still preserving much of the detail, which promises to be useful in further analyses. For this elevation model, the result of processing with window size $\omega = 64$ starts losing detail and may be most suited if a broad overview is intended. It can also be seen that, at this level, the bottom of the largest river shows swapping of gray shades between profile and tangential curvature. This is to be expected because the flow points toward the river in most of the river bed except at the center of the river itself, where the flow points along the river path. Overall, the loss of detail in the bottom is such that most practitioners would likely prefer a smaller window size given the resolution of the DEM. Note, however, that this is a 1-m DEM, and larger window sizes are bound to be useful for DEMs of yet higher resolution.

Fig. 17 shows the corresponding results of processing in ArcGIS. In the top row, slope is the only analysis type that is useful. Key features of the water system are barely visible in the profile and planform processing. In the second row, which corresponds to an effective $\omega = 8$ window size (smoothing with a 6 × 6 window, followed by application of the 3 × 3 window for the curvature computation), the slope representation becomes more pronounced and the profile and planform emerge. Residual noise levels are not as problematic for these data for window sizes for $\omega = 8$ and higher as for the artificial data since these are high-quality data with high vertical resolution. However, the results still show pronounced oscillations in the East–West and North–South directions, which are consistent with the discussion in Section II-E and the observations in Fig. 13.

For the DEM data, it is not possible to subtract a ground truth for visualizing errors on a per-raster-point basis as for artificial data. With no ground truth available, it is harder to confirm which curvatures are correct. However, there is no reason to assume that this region should have predominantly East–West and North–South curvature oscillations. The processing using the IART algorithm does not show any such distortions.

C. Performance

For the performance analysis, a much larger raster of size 5000 × 5000 was used. The processing was done a Dell Mobile Precision 5560 Processor 11th Gen Intel Core i5-11500H @ 2.90GHz with 16.0-GB RAM. While the system has a NVIDIA T1200 w/4GB GPU, the IART code is not implemented to use it. Although ArcGIS can in principle make use of GPU processing, this feature was not used in the evaluation.

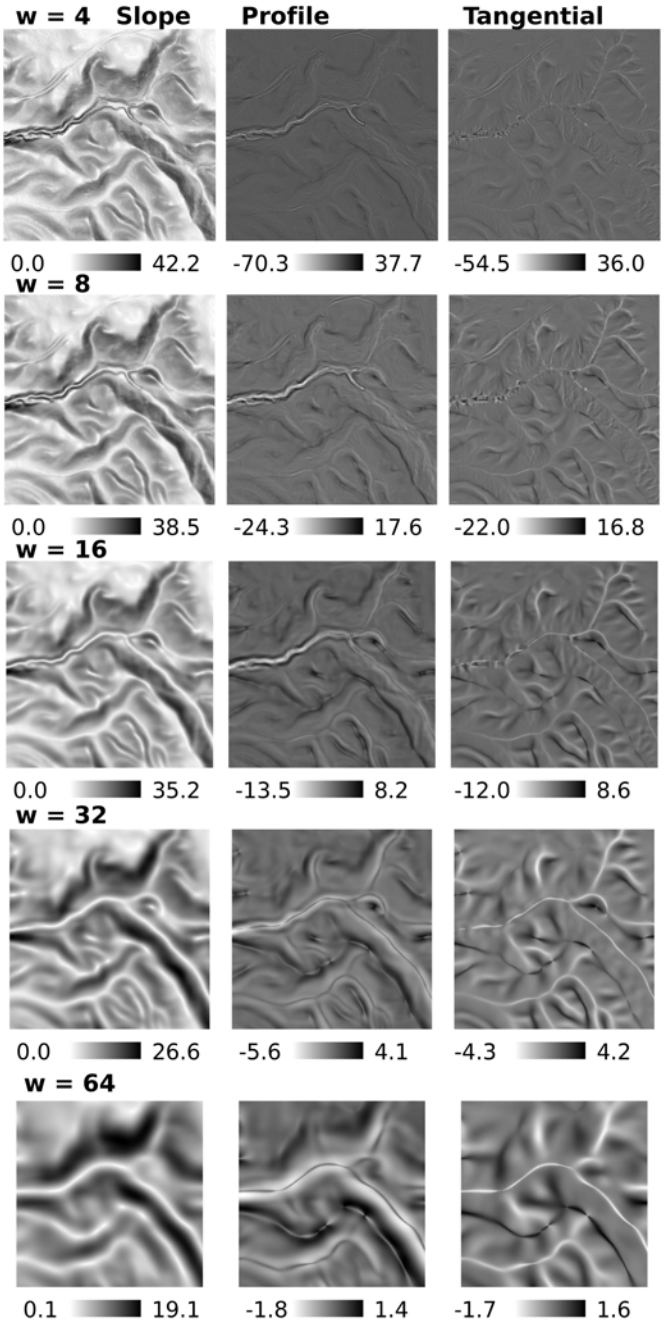

Fig. 16. (Left) Slope, (Middle) profile curvature, and (Right) tangential curvature using the IART algorithm on artificial data with noise, for window sizes (top to bottom) $w = 4$, 8, 16, 32, 64.

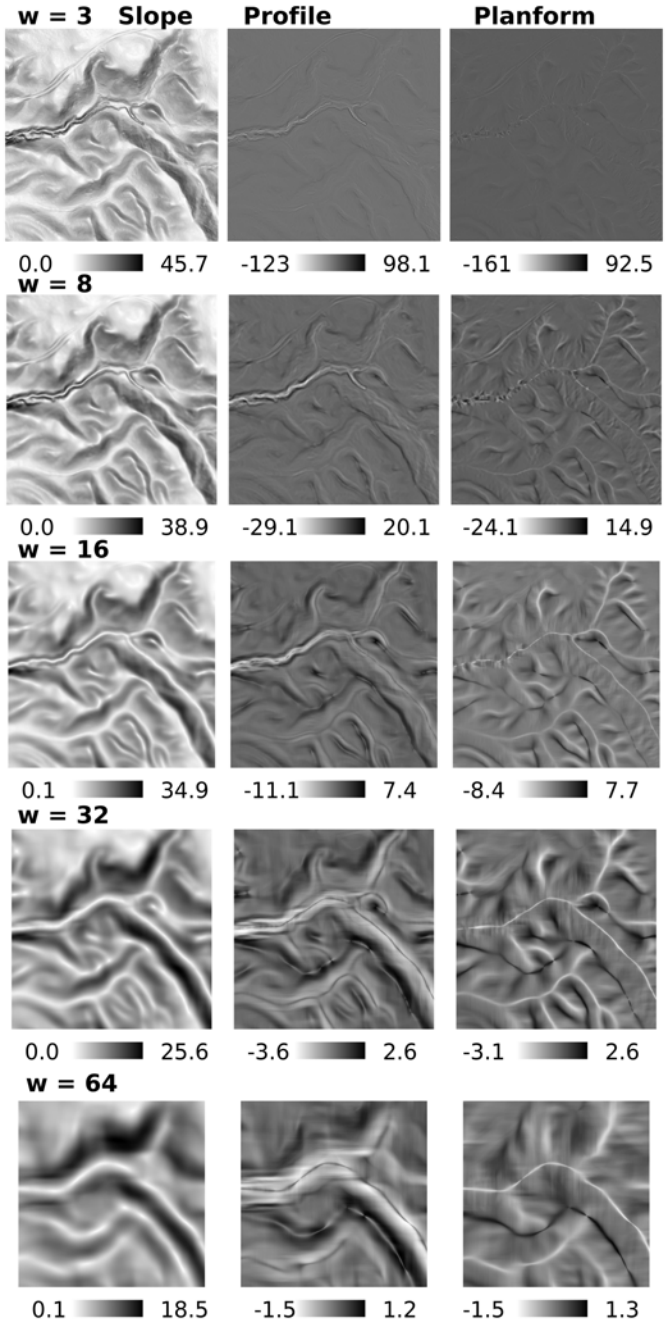

Fig. 17. (Left) Slope, (Middle) profile curvature, and (Right) plan curvature using ArcGIS on the landscape from Fig. 15, for effective window sizes top to bottom 3 (no smoothing), 8, 16, 32, 64 (see Section IV-A).

Fig. 18 shows the contributions to the two types of processing. The largest contributions for both approaches come from the aggregation and focal statistics. The slope and curvature computations are comparatively fast and do not depend on window size. Since profile and tangential/planform require very similar processing times, the two were not distinguished in this figure.

Whether the ArcGIS focal statistics or the IART aggregation is faster and scales more favorably depends to some extent on how they are used. If output at all powers of two of window size is desired, the IART algorithm has constant scaling (black line, triangles up) since each aggregation builds on the previous one. Notice also that these iterations have approximately the same runtime regardless of the level of aggregation. In other words, the shift by $\delta = w/2$ raster points that was discussed in Section III does not affect the runtime negatively. ArcGIS does not offer any such incremental processing. For ArcGIS, the focal statistics (red line, circles) has to be computed from scratch for each window size, and this computation takes longer for window sizes 32 and 64.

If the IART processing is to be done to the final size without employing intermediate aggregation steps, the intermediate

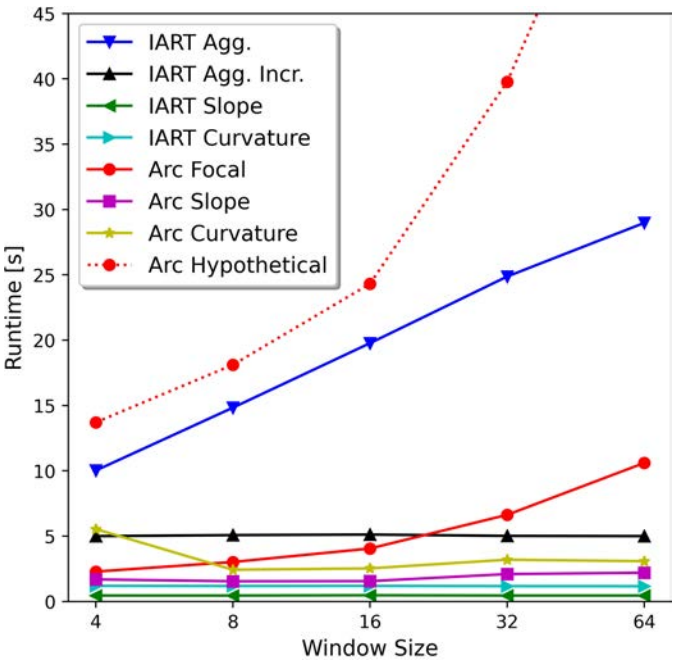

Fig. 18. Aggregation and slope/curvature computation contributions to the runtime for the IART and ArcGIS code for a 5000 × 5000 raster image. See the text for possible combinations.

results are still needed and have to be computed. The resulting processing time is logarithmic in the size of the window, which appears linear in Fig. 18 with its logarithmic x-axis. The IART code was not optimized for this use case, but new memory was allocated in each iteration such that intermediate copies could be preserved. Even without optimizations, the Python-based IART implementation is within the same runtime range as ArcGIS and ArcGIS scale slightly worse. Note that the focal statistics algorithm has to solve the same problem of inherently quadratic scaling. ArcGIS has been optimized over decades and clearly also avoids the quadratic cost.

One could entertain the thought of aggregating the regression terms $\sum xz$, $\sum yz$, $\sum xxz$, $\sum yyz$, and $\sum xyz$ using ArcGIS's focal statistics tool, so as to gain the benefit of a mathematically well-founded regression fit to the data and avoid the approximations of their current two-step pipeline. Such a hypothetical approach would amount to running their focal statistics tool six times, once for $\sum z$ and, in addition, for each of the five regression terms $\sum xz$ and so on (see dotted line "Arc Hypothetical" in Fig. 18). This approach suffers from a poorer scaling of the focal statistics tool. For window size $w = 64$, running the focal statistics tool on all six regression terms would amount to more than twice the runtime of the IART algorithm. However, the benefit in comparison with IART would be that arbitrary window shapes could be used.

Beyond the aggregations and focal statistics, slope and curvature computations contribute to the final result. For the IART algorithm, these take only about 1 s for the 5000 × 5000 image under consideration, which is substantially less than the corresponding computations in ArcGIS. The reason why this portion is faster in the IART algorithm is that the aggregation steps already produce the sums and sums of squares from which the regression coefficients can be calculated directly without a need for further aggregations across raster neighbors. In the ArcGIS approach, the focal statistics step still leaves the need for slope and curvature computations across 3 × 3 windows, which takes more than twice as long.

TABLE III
EXAMPLE SCENARIOS OF HOW RUNTIME CONTRIBUTIONS ARE COMBINED WHEN COMPUTING CURVATURES

Curvatures for all w using IART	Add IART Agg. Incr. and IART Curvature for all w
Curvature for specific w_0 using IART	Add IART Agg. and IART Curvature for w_0
Curvature for specific w_0 using ArcGIS	Add Arc Focal and Arc Curvature for w_0
Curvature for specific w_0 using hypothetical approach	Add Arc Hypothetical and IART Curvature for w_0

Table III shows some example scenarios together with the contributions that would have to be added to estimate the computation time in each case:

In summary, it can be seen that even with an implementation in Python, the overall performance is comparable to ArcGIS for substantially improved output quality, as discussed in Sections IV-A and IV-B. The scaling of the proposed IART algorithm is fully logarithmic in window size, as expected from theoretical considerations, and the output on every length scale is produced as a byproduct of the aggregation process, allowing multiscale analysis at very little additional cost.

V. CONCLUSION AND FUTURE WORK

An algorithm was presented for computing regression-based curvatures over windows of sizes that can be arbitrarily large powers of 2. The definition follows Evans' original idea of fitting quadratic functions to elevation points. Meanwhile, the proposed algorithm avoids the need for encoding specific window sizes that was inherent in previous approaches and is, hence, suitable for modern high-resolution DEM data. Limits to the window size were avoided through an iterative aggregation approach, in which each iteration amounts to a doubling of window size.

The algorithm was evaluated over artificial data that consist of ten hills in the shape of Gaussian functions. Point-based curvatures were computed analytically and used as ground truth. Deviations between window-aggregation results and the ground truth increased in the noise-free setting, as would be expected. Even in that setting, the results of the proposed window-aggregation algorithm were closer to the ground truth than those of the comparison approach of using ArcGIS' focal statistics tool followed by conventional curvature computations.

When random single-pixel noise was added, the match between window-aggregation results and ground truth, as measured by the R^2 value, increased up to the largest window size of 64 × 64 raster points. In other words, larger window sizes consistently resulted in a closer approximation of the analytical point-based curvatures than smaller ones, due to the noise reduction associated with the averaging. These

observations support the value of using larger windows than is possible for typical versions of Evans' regression-based curvature specifications.

A two-step approach of first averaging elevation data and then applying small-window curvature computations also showed a reduction in noise with increasing window size, but the quality was much poorer than for the proposed single-step algorithm that computes regressions over windows directly. This conclusion could be drawn both from a quantitative analysis of the R^2-values with respect to the ground truth and also from an inspection of the difference between individual raster points. While both the proposed, single-step and the comparison, two-step algorithms show the predictable differences between the window-level results and the point-based curvatures, noise in the resulting curvature output makes the comparison approach much less useful. Moreover, in the comparison algorithm, the noise showed horizontal and vertical streaks that are clear evidence of averages being taken over only a small portion of the raster points in the averaging windows. This behavior is expected because all but edge values cancel when differences are taken between two neighboring raster points, each of which represents an average over a window.

The algorithm was furthermore evaluated on 1-m resolution DEM data from the Souris River Basin in North Dakota. This evaluation supports the notion that at such high resolution, the results of curvature computations for small-window sizes are dominated by noise and not useful. While curvatures for window sizes of 4×4 barely show structure, interesting features emerge when window sizes are increased. While averaging also helps in the two-step comparison approach, the results of the evaluation on real data suffer from the same horizontal and vertical streaks as those in the evaluation on artificial data, and they are of noticeably lower quality.

In summary, an approach was presented for computing high-quality curvature rasters from high-resolution DEM data and bypass approximations that are commonly made in this process. While the approach has consistently higher quality than existing ones, the computation time is comparable and the implementation scales slightly better with window size. Generalizing the principles of the proposed algorithm to any of the topographic variables that rely on polynomial fits would be straightforward, and even some other geomorphometric variables satisfy the additivity criterion that underlies its effectiveness. With the increase in availability of elevation data, this work promises to become increasingly relevant for addressing the important challenges of understanding surface hydrology and erosion.

ACKNOWLEDGMENT

The authors would like to thank Alan Denton and Guy Hokanson for proofreading the manuscript and Grit May for pointing them to newly available 1-m resolution digital elevation model (DEM) data.

REFERENCES

[1] X. Liu, "Airborne LiDAR for DEM generation: Some critical issues," *Prog. Phys. Geography*, vol. 32, no. 1, pp. 31–49, Feb. 2008.

[2] A. C. V. Getirana, M.-P. Bonnet, O. C. Rotunno Filho, and W. J. Mansur, "Improving hydrological information acquisition from DEM processing in floodplains," *Hydrol. Processes*, vol. 23, no. 3, pp. 502–514, Jan. 2009.

[3] J. N. Callow, K. P. Van Niel, and G. S. Boggs, "How does modifying a DEM to reflect known hydrology affect subsequent terrain analysis?" *J. Hydrol.*, vol. 332, nos. 1–2, pp. 30–39, Jan. 2007.

[4] I. D. Moore, R. B. Grayson, and A. R. Ladson, "Digital terrain modelling: A review of hydrological, geomorphological, and biological applications," *Hydrol. Processes*, vol. 5, no. 1, pp. 3–30, Jan. 1991.

[5] M. Chang, *Forest Hydrology: An Introduction to Water and Forests*. Boca Raton, FL, USA: CRC Press, 2006.

[6] W. T. Struble and J. J. Roering, "Hilltop curvature as a proxy for erosion rate: Wavelets enable rapid computation and reveal systematic underestimation," *Earth Surf. Dyn.*, vol. 9, no. 5, pp. 1279–1300, 2021.

[7] Q. Wu, Y. Chen, J. P. Wilson, H. Tan, and T. Chu, "A new approach for calculating the slope length factor in the revised universal soil loss equation," *J. Soil Water Conservation*, vol. 76, no. 2, pp. 153–165, 2021.

[8] L. Burian and J. Minár, "Utilization of a comparison of curvatures for land surface segmentation," *Central Eur. J. Geosci.*, vol. 5, no. 4, pp. 560–569, 2013.

[9] B. Romstad and B. Etzelmüller, "Mean-curvature watersheds: A simple method for segmentation of a digital elevation model into terrain units," *Geomorphology*, vols. 139–140, pp. 293–302, Feb. 2012.

[10] I. Evans, "General geomorphometry, derivatives of altitude, and descriptive statistics," in *Spatial Analysis in Geomorphology*. Abingdon, U.K.: Routledge, 1972, pp. 17–90.

[11] J. Schmidt, I. S. Evans, and J. Brinkmann, "Comparison of polynomial models for land surface curvature calculation," *Int. J. Geographical Inf. Sci.*, vol. 17, no. 8, pp. 797–814, Dec. 2003.

[12] L. Blaga, "Aspects regarding the significance of the curvature types and values in the studies of geomorphometry assisted by GIS," Ann. Univ. Oradea, Geography Series/Analele Universitatii din Oradea, Seria Geografie, Oradea, Romania, 2012, vol. 22, no. 2, Art. no. 222116-599.

[13] I. V. Florinsky, "An illustrated introduction to general geomorphometry," *Prog. Phys. Geography*, vol. 41, no. 6, pp. 723–752, Dec. 2017.

[14] J. Minár, I. S. Evans, and M. Jenčo, "A comprehensive system of definitions of land surface (topographic) curvatures, with implications for their application in geoscience modelling and prediction," *Earth-Sci. Rev.*, vol. 211, Dec. 2020, Art. no. 103414.

[15] J. Minár, M. Jenčo, and I. S. Evans, "What does land surface curvature really mean?" in *Proc. Geomorphometry Conf.*, M. Alvioli, I. Marchesini, L. Melelli, and P. Guth, Eds., 2020, doi: 10.30437/GEOMORPHOMETRY2020_6.

[16] X. Liu and Z. Zhang, "LiDAR data reduction for efficient and high quality DEM generation," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 37, pp. 173–178, Jul. 2008.

[17] M. Meadows and M. Wilson, "A comparison of machine learning approaches to improve free topography data for flood modelling," *Remote Sens.*, vol. 13, no. 2, p. 275, Jan. 2021.

[18] C. W. T. Yeu, M.-H. Lim, G.-B. Huang, A. Agarwal, and Y.-S. Ong, "A new machine learning paradigm for terrain reconstruction," *IEEE Geosci. Remote Sens. Lett.*, vol. 3, no. 3, pp. 382–386, Jul. 2006.

[19] J. Lee, J. Im, K. Kim, and L. Quackenbush, "Machine learning approaches for estimating forest stand height using plot-based observations and airborne LiDAR data," *Forests*, vol. 9, no. 5, p. 268, May 2018.

[20] J. S. Evans and A. T. Hudak, "A multiscale curvature algorithm for classifying discrete return LiDAR in forested environments," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 4, pp. 1029–1038, Apr. 2007.

[21] H. Kim, J. Willers, and S. Kim, "Digital elevation modeling via curvature interpolation for LiDAR data," *Electron. J. Differ. Equ.*, vol. 23, pp. 47–57, Jan. 2016.

[22] Y. Pan, J. Xia, and K. Yang, "A method for digital terrain reconstruction using longitudinal control lines and sparse measured cross sections," *Remote Sens.*, vol. 14, no. 8, p. 1841, Apr. 2022.

[23] D. Thibault and C. M. Gold, "Terrain reconstruction from contours by skeleton construction," *GeoInformatica*, vol. 4, no. 4, pp. 349–373, 2000.

[24] I. S. Evans, "Land surface derivatives: History, calculation and further development," in *Proc. Geomorphometry*, 2013, pp. 16–20.
[25] J. P. Rigol-Sanchez, N. Stuart, and A. Pulido-Bosch, "ArcGeomorphometry: A toolbox for geomorphometric characterisation of DEMs in the ArcGIS environment," *Comput. Geosci.*, vol. 85, pp. 155–163, Dec. 2015.
[26] I. S. Evans, "An integrated system of terrain analysis and slope mapping," *Zeitschrift für Geomorphologie. Supplementband Stuttgart*, vol. 36, pp. 274–295, Jan. 1980.
[27] J. Wood, "The geomorphological characterisation of digital elevation models," Ph.D. dissertation, Dept. Geography, Univ. Leicester, Chennai, Tamil Nadu, 1996.
[28] J. Wood, "Modelling the continuity of surface form using digital elevation models," in *Proc. 8th Int. Symp. Spatial Data Handling*, Vancouver, BC, Canada, 1998, pp. 725–736.
[29] M. Albani, B. Klinkenberg, D. W. Andison, and J. P. Kimmins, "The choice of window size in approximating topographic surfaces from digital elevation models," *Int. J. Geographical Inf. Sci.*, vol. 18, no. 6, pp. 577–593, 2004.
[30] (2021). *Focal Statistics*. [Online]. Available: https://desktop.arcgis.com/en/arcmap/10.3/tools/spatial-analyst-toolbox/focal-statistics.htm
[31] J. Minár and I. S. Evans, "Towards exactness in geomorphometry," *Genesis*, vol. 166, no. 167, p. 168, 2015.
[32] J. Lee, P. Fisher, and P. Snyder, "Modeling the effect of data errors on feature extraction from digital elevation models," *Photogramm. Eng. Remote Sens.*, vol. 58, p. 1461, Feb. 1992.
[33] M. P. Kumler, "An intensive comparison of triangulated irregular networks (TINs) and digital elevation models (DEMs)," *Cartographica*, vol. 31, no. 2, pp. 1–99, Jun. 1994.
[34] P. V. Bolstad and T. Stowe, "An evaluation of DEM accuracy: Elevation, slope, and aspect," *Photogramm. Eng. Remote Sens.*, vol. 60, no. 11, pp. 7327–7332, 1994.
[35] S. D. Warren, M. G. Hohmann, K. Auerswald, and H. Mitasova, "An evaluation of methods to determine slope using digital elevation data," *CATENA*, vol. 58, no. 3, pp. 215–233, Dec. 2004.
[36] M. Ghandehari, "Cross-scale analysis of surface-adjusted measurements in digital elevation models," Ph.D. dissertation, Dept. Geography, Univ. Colorado Boulder, Boulder, CO, USA, 2019.
[37] J. Khanifar and A. Khademalrasoul, "Multiscale comparison of LS factor calculation methods based on different flow direction algorithms in Susa ancient landscape," *Acta Geophysica*, vol. 68, no. 3, pp. 783–793, Jun. 2020.
[38] J. Khanifar and A. Khademalrasoul, "Effects of neighborhood analysis window forms and derivative algorithms on the soil aggregate stability–landscape modeling," *Catena*, vol. 198, Mar. 2021, Art. no. 105071.
[39] P. A. Shary, L. S. Sharaya, and A. V. Mitusov, "The problem of scale-specific and scale-free approaches in geomorphometry," *Geografia Fisica e Dinamica Quaternaria*, vol. 28, no. 1, pp. 81–101, 2005.
[40] J. Gao, "Impact of sampling intervals on the reliability of topographic variables mapped from grid DEMs at a micro-scale," *Int. J. Geographical Inf. Sci.*, vol. 12, no. 8, pp. 875–890, Dec. 1998.
[41] J. Garbrech and L. Mart, "Grid size dependency of parameters extracted," *Comput. Geosci.*, vol. 20, no. 1, pp. 85–87, 1994.
[42] S. Kienzle, "The effect of DEM raster resolution on first order, second order and compound terrain derivatives," *Trans. GIS*, vol. 8, no. 1, pp. 83–111, Jan. 2004.
[43] R. Srinivasan and B. A. Engel, "Effect of slope prediction methods on slope and erosion estimates," *Appl. Eng. Agricult.*, vol. 7, no. 6, pp. 779–783, 1991.
[44] M. Dunn and R. Hickey, "The effect of slope algorithms on slope estimates within a GIS," *Cartography*, vol. 27, no. 1, pp. 9–15, Jun. 1998.
[45] L. Xiong, G. Tang, X. Yang, and F. Li, "Geomorphology-oriented digital terrain analysis: Progress and perspectives," *J. Geographical Sci.*, vol. 31, no. 3, pp. 456–476, Mar. 2021.
[46] H. Mitášová and J. Hofierka, "Interpolation by regularized spline with tension: II. Application to terrain modeling and surface geometry analysis," *Math. Geol.*, vol. 25, no. 6, pp. 657–669, 1993.
[47] M. Ligas and P. Banasik, "Conversion between Cartesian and geodetic coordinates on a rotational ellipsoid by solving a system of nonlinear equations," *Geodesy Cartography*, vol. 60, no. 2, pp. 145–159, Jan. 2011.
[48] J. Berry, "Beyond mapping use surface area for realistic calculations," *Geo World*, vol. 15, pp. 20–21, Mar. 2002.
[49] P. A. Strobl *et al.*, "The digital elevation model intercomparison experiment demix, a community-based approach at global DEM benchmarking," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 43, pp. 395–400, Jun. 2021.
[50] A. M. Denton, M. Ahsan, D. Franzen, and J. Nowatzki, "Multi-scalar analysis of geospatial agricultural data for sustainability," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 2139–2146.
[51] A. M. Denton, R. Gomes, and D. Franzen, "Scaling up window-based slope computations for geographic information system," in *Proc. IEEE Int. Conf. Electro/Inf. Technol. (EIT)*, May 2018, pp. 0554–0559.
[52] R. Gomes, A. Denton, and D. Franzen, "Quantifying efficiency of sliding-window based aggregation technique by using predictive modeling on landform attributes derived from DEM and NDVI," *ISPRS Int. J. Geo-Inf.*, vol. 8, no. 4, p. 196, Apr. 2019.
[53] J. Krcho, "Georelief as a subsystem of landscape and the influence of morphometric parameters of georelief on spatial differentiation of landscape-ecological processes," *Ekologia*, vol. 10, no. 2, pp. 115–157, 1991.
[54] L. W. Zevenbergen and C. R. Thorne, "Quantitative analysis of land surface topography," *Earth Surf. Processes Landforms*, vol. 12, no. 1, pp. 47–56, Jan. 1987.
[55] J. Minár *et al.*, "Third-order geomorphometric variables (derivatives): Definition, computation and utilization of changes of curvatures," *Int. J. Geographical Inf. Sci.*, vol. 27, no. 7, pp. 1381–1402, Jul. 2013.
[56] S. E. Franklin, "Interpretation and use of geomorphometry in remote sensing: A guide and review of integrated applications," *Int. J. Remote Sens.*, vol. 41, no. 19, pp. 7700–7733, Oct. 2020.
[57] G. Hu, W. Dai, S. Li, L. Xiong, and G. Tang, "A vector operation to extract second-order terrain derivatives from digital elevation models," *Remote Sens.*, vol. 12, no. 19, p. 3134, Sep. 2020.
[58] I. S. Evans and J. Minár, "A classification of geomorphometric variables," in *Geomorphometry*. Redlands, CA, USA, 2011, pp. 105–108.
[59] P. Shary, "The second derivative topographic method," in *The Geometry of the Earth Surface Structures*. Pushchino, Russia: Pushchino Research Centre Press, 1991, pp. 30–60.
[60] I. V. Florinsky, "Accuracy of local topographic variables derived from digital elevation models," *Int. J. Geographical Inf. Sci.*, vol. 12, no. 1, pp. 47–62, Jan. 1998.
[61] E. W. Weisstein. (2005). *Power Sum. From Mathworld—A Wolfram Web Resource*. [Online]. Available: http://mathworld.wolfram.com/PowerSum.html
[62] Q. Wu, Y. Chen, J. P. Wilson, X. Liu, and H. Li, "An effective parallelization algorithm for DEM generalization based on CUDA," *Environ. Model. Softw.*, vol. 114, pp. 64–74, Apr. 2019.
[63] D. O. W. Resources and N. Dakota. (2017). *QA Program for High Resolution Lidar Data, Multiple Counties in North Dakota, Phase 7, LIDAR_JameRiver_Ph10_QL2, Block Ward*. [Online]. Available: https://lidar.dwr.nd.gov

Anne M. Denton (Member, IEEE) received the M.S. degree in computer science from North Dakota State University (NDSU), Fargo, ND, USA, in 2003, and the Ph.D. degree in physics from Johannes Gutenberg University, Mainz, Germany, in 1996.

She is currently a Professor with the Department of Computer Science, NDSU. She has been involved in several interdisciplinary research projects related primarily to bioinformatics and agriculture. She has published more than 70 peer-reviewed journal and conference publications and has led projects funded at a total of more than one million dollars. Her research interests are in data science of large complex datasets, with a recent focus on geospatial data.

Rahul Gomes (Member, IEEE) received the Ph.D. degree in computer science from North Dakota State University, Fargo, ND, USA, in 2019, with a focus on optimizing sliding window-based algorithms for deriving topological variables used in geographical information systems.

He is currently an Assistant Professor of computer science with the University of Wisconsin–Eau Claire, Eau Claire, WI, USA. His research interests pertain to application of machine learning algorithms to process big data in the field of bioinformatics and geospatial analysis. A significant portion of this research employs deep learning and high throughput computing. He enjoys teaching computer science courses and fostering the spirit of student–faculty collaborative research on campus.

David M. Schwartz grew up in Bismarck, ND, USA. He received the bachelor's degree in computer science and mathematics and the master's degree in computer science, studying with Dr. Anne M. Denton, from North Dakota State University, Fargo, ND, in 2020 and 2021, respectively.

He moved to Colorado to work on aerospace software and now works on medically device software in the same area. He is interested in algorithmic and graphical applications of computers.

David W. Franzen received the B.S. degree in forestry, the M.S. degree in soil fertility, and the Ph.D. degree in soil chemistry from the University of Illinois Urbana-Champaign, Champaign, IL, USA, in 1975, 1976, and 1993, respectively.

He worked 18 years as agronomist/manager for chain of retail fertilizer and ag-supply company in Illinois. He is currently a Professor of soil science and an Extension Soil Specialist with North Dakota State University (NDSU), Fargo, ND, USA. He joined NDSU as a Faculty Member in 1994. His research focuses on site-specific crop nutrient management and the revision of fertilizer recommendations to support it.

Dr. Franzen is a fellow of American Society of Agronomy.