EPiC Series in Computing

Volume 104, 2024, Pages 14–24

Proceedings of 3rd International Workshop on
Mathematical Modeling and Scientific Computing

Stabilizing Tensor Voting for 3D Curvature Estimation

Victor Alfaro Pérez[1], Virginie Uhlmann[2], and Brigitte Forster-Heinlein[1]

[1] University of Passau, Faculty of Computer Science and Mathematics, Passau, Germany
vicalpeg@gmail.com, brigitte.forster@uni-passau.de
[2] European Bioinformatics Institute (EMBL-EBI), Cambridge, UK
uhlmann@ebi.ac.uk

Abstract

Curvature plays an important role in the function of biological membranes, and is therefore a readout of interest in microscopy data. The PyCurv library established itself as a valuable tool for curvature estimation in 3D microscopy images. However, in noisy images, the method exhibits visible instabilities, which are not captured by the standard error measures. In this article, we investigate the source of these instabilities, provide adequate measures to detect them, and introduce a novel post-processing step which corrects the errors. We illustrate the robustness of our enhanced method over various noise regimes and demonstrate that with our orientation correcting post-processing step, the PyCurv library becomes a truly stable tool for curvature quantification.

1 Introduction

The development of 3D microscopy modalities, such as cryo-electron tomography, is providing valuable insights into the intricate geometry of biological objects at the cellular and intracellular level. The shape of the membrane of cells, organelles, or vesicles, quantified by their curvature, is of particular interest as it informs on underlying structural and functional properties [3, 5]. However, accurate curvature estimation from large, information-rich microscopy image volumes poses a significant challenge due to acquisition and sample preparation artifacts, often leading to incomplete pictures of the cell membrane surface [11]. Missing or corrupted data points can hinder both quantitative and qualitative curvature estimation. To address this problem, robust methods that can handle noisy and incomplete image volume data are needed.

Various strategies have been proposed for estimating curvature from image data. These include analytical, discrete, and tensor voting (TV) approaches. Analytical methods involve fitting surfaces or curvature tensors using mathematical models [2, 4, 10]. Discrete methods estimate the curvature at each vertex or face of a mesh using discretized formulas of the differential geometry of surfaces [8, 13]. While these methods have proven successful in certain scenarios, they are not robust enough to handle the complicated and imperfect nature of 3D microscopy data. In contrast, TV methods have emerged as a robust approach for analyzing curvature in the presence of noise and for dealing with the kind of missing or irregular data found in bioimages [9, 14, 16]. These approaches accumulate local shape information from the data to better approximate curvature.

State-of-the art results in terms of accuracy and robustness to noise for curvature estimation in microscopy image volumes are currently achieved by Augmented Vector Voting (AVV) as implemented in the PyCurv software [11]. The AVV method has been tested against well-established contenders on benchmark surfaces and biological data, demonstrating its superior performance. Nevertheless, we identified some limitations in PyCurv's AVV method when dealing with different types of noise added on benchmark surfaces. Estimated normals tend to be erroneously oriented, leading to incorrect curvature estimation. To overcome this limitation, we here introduce a novel orientation correcting post-processing step after the estimation of normals carried out by PyCurv's AVV method. The orientation correction step corrects the erroneously refined normals, ensuring a consistent orientation and improving the accuracy of the curvature estimation results. Our proposed approach is demonstrated to significantly improve the accuracy and robustness of curvature estimation in noisy data. Ultimately, our novel orientation correcting step enhances the robustness of PyCurv's AVV method by ensuring a consistent performance even in the presence of noise, thus broadening the range of applications in which the method performs well.

The paper is structured as follows. In Section 2, we provide the relevant technical background on tensor voting and discuss related works. In Section 3, we describe the details of the AVV algorithm as implemented in PyCurv, and introduce our novel orientation correcting step. We present experimental results in Section 4 and close the paper with concluding remarks in Section 5.

2 Background on tensor voting

TV methods have recently gained importance in curvature analysis due to their ability to mitigate the effects of noise and missing information by relying on the accumulation of neighboring information [9, 16]. TV encodes geometrical information of each input point as a second-order symmetric tensor, which can be geometrically represented as an ellipsoid whose shape reflects the encoded feature, while its size indicates the likelihood of the point belonging to that feature. The core of TV lies in the communication of information between neighboring points to refine the initial tensors and thereby enhance the geometrical information encoded at each input point. Each point receives votes from neighboring points within a predefined neighborhood. The voting process is carried out by specially designed voting fields for each tensor component [6]. Through this refinement, the encapsulated geometrical information becomes more precise. Depending on the objectives of the analysis, various approaches can be pursued, such as dense extrapolation to infer features at new locations over the 3D space [7], feature extraction by relating the final refined tensors to overall features [14], or identifying outliers or points of interest, such as corners or discontinuities [9].

Curvature information has been incorporated into TV in several ways. Tang and Medioni [14] introduced an approach to classify input points as planar, elliptic, parabolic, or hyperbolic and estimate principal curvatures. Their method is however primarily aimed at extracting the direction of principal curvatures, not their magnitude. Taubin [15] estimates curvature information at each vertex of a triangular mesh approximating a surface. This method provides only a single scale of analysis and doesn't robustly give the terms κ_{ij} and T_{ij} in the formula that approximates the matrix that yields the principal directions and curvatures at a centroid c_i, given by

$$M_{c_i} = \sum_{c_j \in V^i} w_{ij} \kappa_{ij} T_{ij} T_{ij}^t. \tag{1}$$

	Tang and Medioni [14]	**Taubin** [15]	**Page et al.** [9]	**Tong & Tang** [16]	**Salfer et al. (PyCurv)** [11]
Input	Unstructured cloud of points or mesh	Mesh	Mesh	Unstructured cloud of points or mesh	Mesh mapped onto a graph
Discontinuity detection	–	–	✓	✓	✓
Position Correction	–	–	–	✓	–
Normal estimation	Tensor voting	Normalized weighted sum of the normals of the incident faces	Tensor voting weighted by triangle area	Tensor voting	Tensor voting weighted by triangle area
Votes casted by	Input points	Triangle vertices	Triangle centroids	Extracted points at a specified distance	Triangle faces
Votes collected at	Input points	Triangle vertices	Triangle vertices	Detected inliers	Triangle faces
Voting neighbourhood	Defined by Euclidean distance	1-ring neighbourhood	Defined by a geodesic distance	Defined by a RadiusHit	Defined by a geodesic distance depending on a RadiusHit
Scale	Free parameter	Single Scale	Free parameter	Adaptive	Adaptive
Output	Principal directions	Principal curvatures and principal directions	Principal curvatures and principal directions	Principal curvatures and principal directions	Principal curvatures and principal directions
Robust to noise	✓	–	✓	✓	✓

Table 1: Comparative analysis of tensor voting curvature estimation methods.

Here, M_{c_i} represents the matrix at c_i, V^i is the set of centroids within the geodesic neighborhood of c_i, T_{ij} is the normalized projection of the vector $c_i c_j$ onto the tangent plane of c_i, and κ_{ij} is the curvature of the circular arc connecting c_i and c_j. Page et al. [9] extended these techniques with the introduction of Normal Tensor Voting, which allowed to consider discontinuities on the analyzed surface and provided multiscale analysis. Other authors such as Tong and Tang [16] have further improved the ability to incorporate more flexible forms of input such as unordered point sets. Salfer et al. [11] have made significant progress in the TV framework with PyCurv by introducing a mapping of the initial triangle mesh to a particular graph that efficiently allows for the computation of geodesic distances and encapsulates discontinuity information. We summarize these methods in Table 1.

A limiting factor of TV is that it requires well-oriented surfaces to perform optimally, which can be difficult to achieve in real applications due to noise and segmentation masks with holes. PyCurv's proposed workflow partially addresses this problem by relying on filled segmentation masks. This is however not done as a built-in part of the TV method itself, but rather as a preprocessing step.

3 Methods

3.1 Augmented vector voting

PyCurv's AVV method operates on the centroids of a triangular mesh, where the center of each triangular face in the input mesh is initially aligned with each corresponding face normal. Mesh extraction involves first applying a fill operator to the initial segmentation mask and then running a marching cubes algorithm to extract a mesh from the filled segmentation. In the tensor refinement stage, each triangle centroid is associated with a refined normal, allowing the subsequent curvature estimation. PyCurv relies on a manually-set parameter to set the size of the region from which curvature votes will be collected. This allows adaptation to different feature scales.

The estimation of curvature is facilitated by (1), which is a discretized version of a symmetric matrix calculated through an integral formula that encapsulates the principal directions and curvatures at a particular point. The weights w_{ij} in (1) are defined using an exponential decay function, assigning higher values to centroids closer to c_i. The eigenvectors of the matrix M_{c_i} yield the estimated principal directions and the eigenvalues allow for calculating the principal curvatures at c_i [15].

3.2 Improved accuracy through orientation correction

The accuracy of the surface curvature estimation is highly dependent on the consistent global orientation of surface normals. While PyCurv's AVV preprocessing workflow aims to close holes and restore orientation before applying TV, it does not fix everything: in areas where the initial normals are almost perpendicular to the true normals, the orientation estimates are generally poor. This leads to incorrect orientations after normal correction, as depicted in Fig. 1 (center).

Erratic normals do not only occur on simple test surfaces, but also appear on sophisticated test bodies and real microscopy data, as pictured in Figs. 2 and 3. Hence, the measurement of curvature data for all these surfaces can be affected.

To address this issue, we have designed an orientation correcting (OC) post-processing step that corrects normal orientation errors and thus improves the accuracy of curvature estimation. The OC step considers each refined normal and its surroundings within a geodesic neighborhood

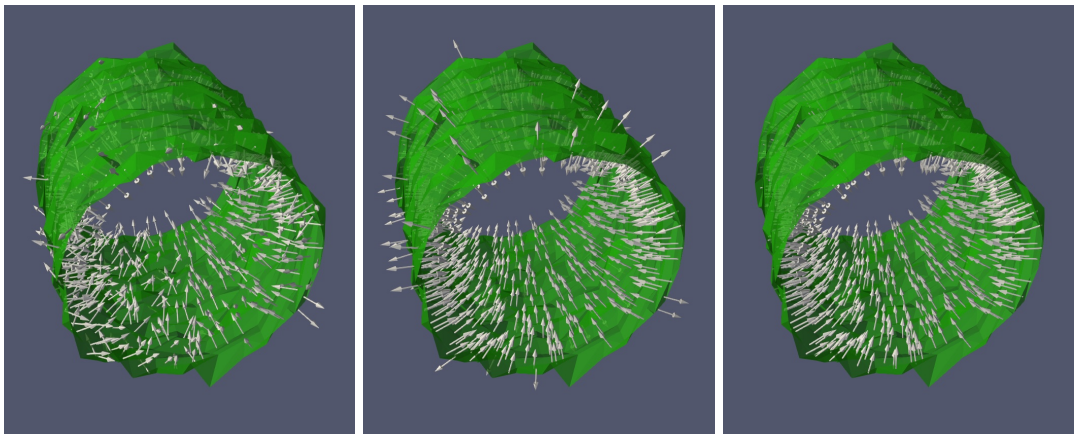

Figure 1: Our proposed post-processing step accurately corrects erroneous normal orientations. Left: Triangulated cylinder with 25% of Gaussian noise added in random directions. Some normals erroneously point outwards. Center: Normals corrected with AVV still exhibit this inconsistency. Right: Our orientation correction (OC) step yields correct normals.

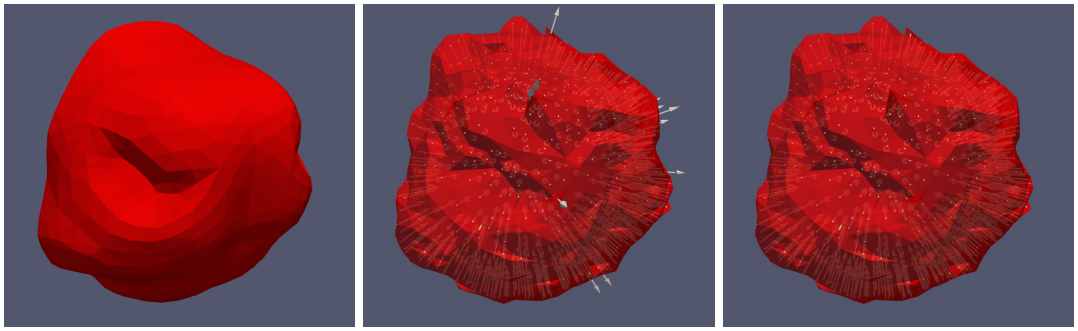

Figure 2: Normal estimation on a smooth cow's mouth surface from https://gitlab.kitware.com/vtk/vtk-examples. Left: reference surface. Center: Same surface perturbed by 5% additive random Gaussian noise. Several of the normals estimated with AVV incorrectly point outward, showing orientation inconsistency. Right: Our orientation correction (OC) step provides correct normals.

of the same size as the voting neighborhood. We then calculate the Euclidean inner (dot $\cdot$) product of each estimated normal N with the average refined neighboring normals N_{avg}. A negative dot product indicates a discrepancy in the orientation of N from its neighboring normals. Its orientation is thus corrected by reversing its sign via the following formula:

$$N_{\text{corrected}} = \begin{cases} N & \text{if } N \cdot N_{\text{avg}} \geq 0, \\ -N & \text{if } N \cdot N_{\text{avg}} < 0. \end{cases}$$

Our OC step offers a simple yet principled solution for global orientation errors in surface normals and thus for improved curvature estimation. While not computationally inexpensive, it remains significantly lighter than the actual voting process. This is a reasonable trade-off given the significant improvement in accuracy it provides. The underlying idea is remarkably simple

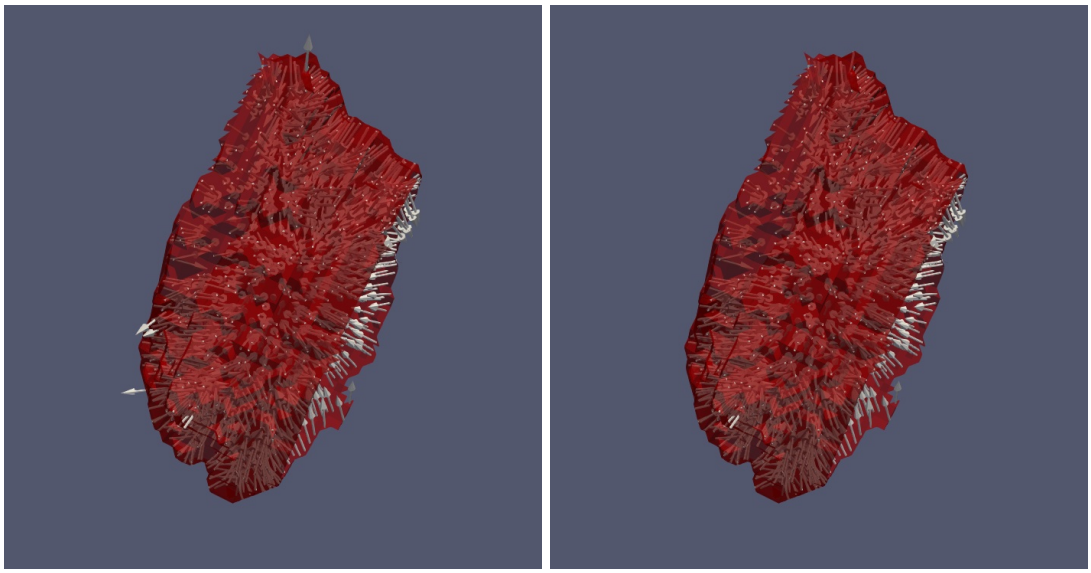

Figure 3: Normal estimation on a human HeLa cell from the PyCurv repository https://github.com/kalemaria/pycurv/tree/master/experimental_data_sets/vesicle. Left: applying PyCurv off-the-shelf results in several wrongly oriented normals. Right: our orientation correcting step appropriately corrects erroneous normals.

to implement, yet remains flexible enough to adjust the size of the reference neighborhood to the features of interest. The OC step is also very robust: as long as a local set of normals has a correct orientation, the OC step is unlikely to fail (Fig. 1).

4 Results

To evaluate the benefit of our OC step towards reducing errors in normal and principal curvature estimation, we establish an experimental setup that takes the following scenarios into account.

a) Noise conditions. We perturb the triangular meshes by adding noise either in the direction of the normal vector to each triangular face or randomly. We increasingly apply Gaussian noise based on a specified percentage of the average edge length of the triangles constituting the surfaces, by moving each triangle along either a predetermined or a random direction.

b) Mesh resolution. Triangular meshes are controlled by a resolution parameter, which determines their level of detail. Higher mesh resolutions are expected to improve the performance of TV, as more neighborhood information is available. Here, we generate different triangulations, including irregular and minimal ones, allowing us to explore a wide range of surface variations and configurations.

c) Reference surfaces: Since true normals are not accessible in noisy surfaces, we use a reference smooth surface as ground truth. The normal of each centroid in the triangle mesh is then compared to the normal of its nearest point on the ground truth smooth surface.

To assess performance, we use the relative error measure given by

$$\Delta_{\overline{e}} = (\overline{e}_{\text{initial}} - \overline{e}_{\text{corrected}})/\overline{e}_{\text{initial}}, \tag{2}$$

where $\overline{e}_{\text{initial}}$ denotes the mean error in normals before applying TV, and $\overline{e}_{\text{corrected}}$ after correction. A value of 1 indicates that the error in normal estimation has been entirely corrected.

Although we are ultimately interested in curvature, considering its relation to the correctness of the normals estimation is useful as curvature captures the rate of change of the normals and is thus highly sensitive to noise. To measure both the initial and corrected errors in normal estimation, we compare them against their reference ground truth. For principal curvatures we use the absolute error measure

$$\Delta\kappa = |\kappa_t - \kappa_e|,$$

where κ_t is the true and κ_e the estimated curvature. Smaller values indicate more accurate estimations. For normal directions, given by unit vectors, in [11] the error measure

$$\Delta v = 1 - |v_t \cdot v_e|,$$

is considered, where v_t is the true and v_e the estimated vector. It returns 0 if the vectors are parallel and 1 if they are perpendicular. This measure cannot distinguish the orientation of the vectors.

To assess the orientation of the normals, we introduce a new measure that calculates the angular error between vectors in radians as

$$\Delta\alpha = \arccos(v_t \cdot v_e).$$

Based on this, we define two relative error measures: the *Vector Error Reduction* Δ_{vector} and the *Angular Vector Error Reduction* Δ_{angle}, defined respectively as

$$\Delta_{\text{vector}} = (\Delta v_{\text{initial}} - \Delta v_{\text{corrected}})/\Delta v_{\text{initial}}, \tag{3}$$

$$\Delta_{\text{angle}} = (\Delta\alpha_{\text{initial}} - \Delta\alpha_{\text{corrected}})/\Delta\alpha_{\text{initial}}. \tag{4}$$

The measure Δ_{angle} is considered superior to Δ_{vector} because Δ_{angle} takes into account the sign of the inner (dot) product between the estimated and the ground truth vectors. When noise is added in random directions, some refined normals may indeed point outward, resulting in inconsistent global orientation. Δ_{angle} accounts for this issue, providing a more accurate measure of result quality. We choose to also report Δ_{vector} as it is based on the error measure Δv originally used in [11] to assess PyCurv's performance. A good correction of the normals according to Δ_{vector} does not however necessarily lead to a good curvature estimation. Misoriented normals were not captured in the original assessment and no robust consistency between small errors in normals and small errors in curvatures was originally observed upon the addition of random noise.

4.1 Experiment 1: performance of the orientation correcting step

The benchmark surfaces used in our experiments were constructed using the VTK Python library [12] and vizualised with the PyCurv Python package and Paraview [1]. They consisted of various geometric shapes, e.g. cylinders, planes, spheres, and spheres with holes. Each benchmark surface was defined by an increasing resolution parameter determining the detail

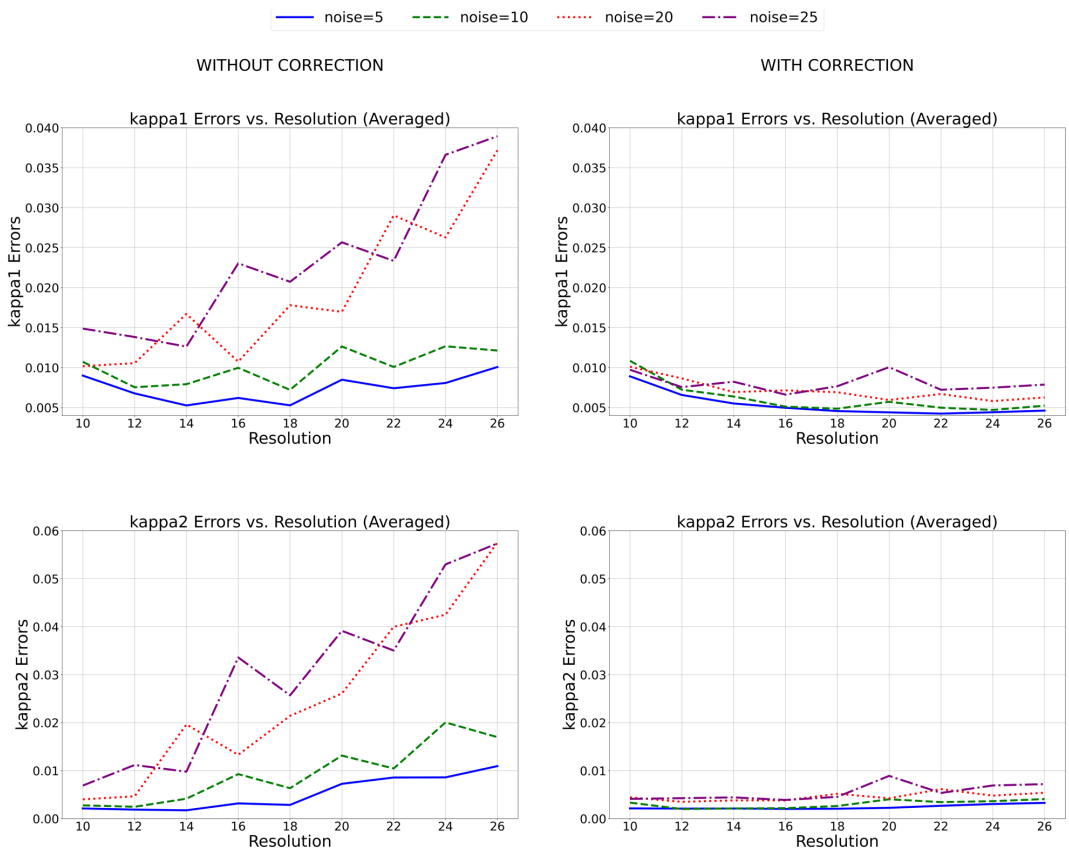

Figure 4: Evolution of the error $\Delta\kappa$ of the principal curvatures κ_1 and κ_2, with increasing mesh resolution on a triangulated sphere. Without correction (left column) the estimated curvature errors diverge with increasing noise level, even for increased resolution. With our orientation correcting OC step (right column), both curvature errors become very small.

level in its discretization and was further triangulated. To introduce noise variations into the surfaces, we increasingly applied Gaussian noise based on a specified percentage of the average edge length of the triangles constituting the surfaces. We did this by moving each triangle along either a predetermined or a random direction.

To measure the accuracy of estimated normals, we used a reference smooth surface as ground truth and compared the normal vectors of each centroid in the triangle mesh of the noisy surface to the normal vector of the nearest point on the smooth surface.

Our OC step is experimentally observed to correct orientation errors and improve the accuracy of curvature estimation, particularly for cases where the AVV method previously failed as seen in Figs. 4 and 5. Both error reduction measures Δ_{vector} and Δ_{angle} offer the same insights since the OC step aims at correcting the missorientation errors of the normals.

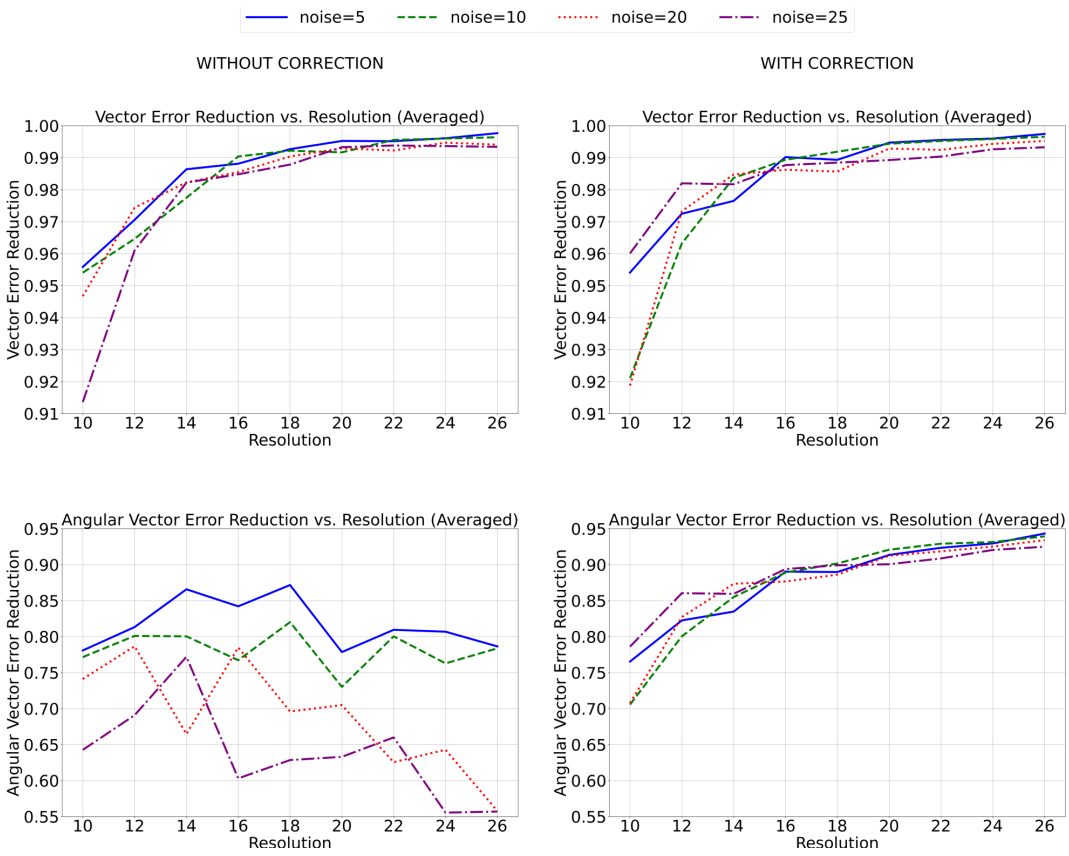

Figure 5: Evolution of the error reduction measures Δ_{vector} and Δ_{angle} with increasing mesh resolution on a triangulated sphere. Without correction (left column), Δ_{vector} approaches the best value of 1, but Δ_{angle} decrease for high noise levels. With our orientation correcting OC step (right column), both relative error measures approach 1 as intended.

4.2 Experiment 2: significance of the orientation correcting step

In our second experiment, we examined the influence of noise, if the mesh resolution is increased, and with this the edge length of the mesh is reduced. In practice it is desirable to have finer meshes that capture more details and therefore allow for a more precise curvature estimation. However, we have observed that the number or misoriented normals increases with higher resolutions, thus augmenting the errors. Figure 6 shows our measurements for finer and finer tessellations of a plane reference surface. The finer the triangulations, the more misoriented normals appear.

5 Conclusion

We proposed a novel post-processing step to correct the normal orientations on triangulated surfaces prior to curvature estimation. To measure performance, we introduced the Vector Error Reduction and the Angular Vector Error Reduction measures. Our experiments showed that

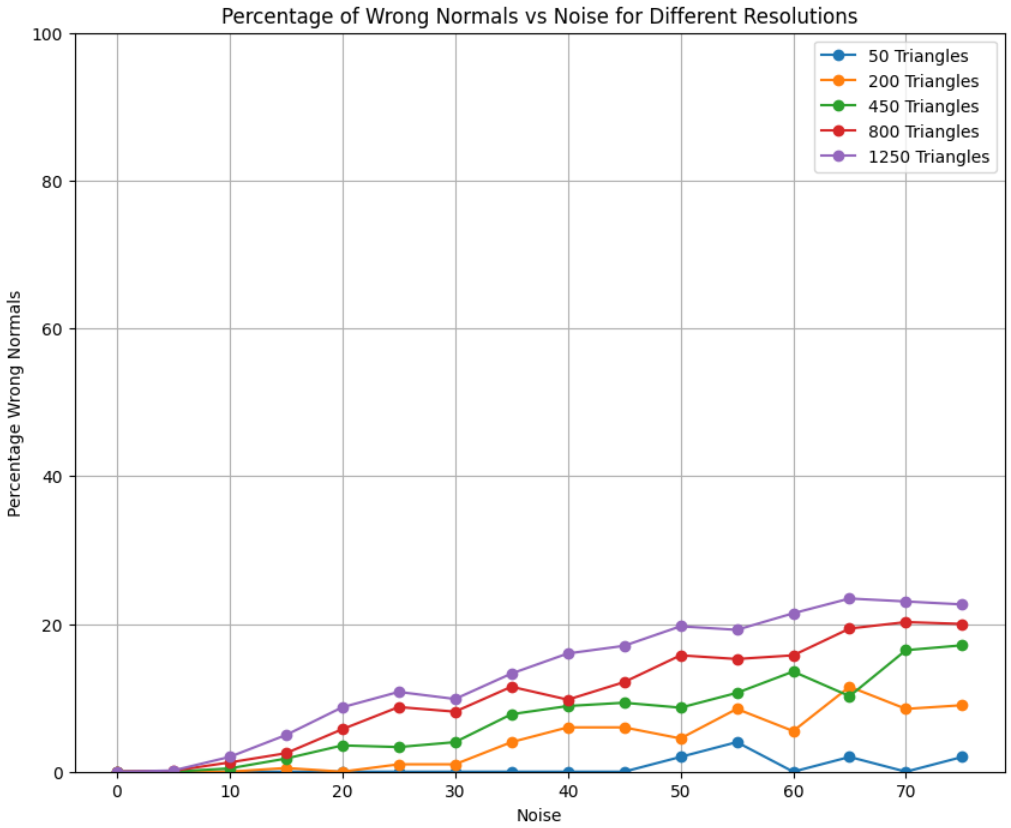

Figure 6: Impact of mesh resolution on normal estimation error. Measurements were taken on a plane of fixed size that has been uniformly tessellated with different resolutions. Our results show that the smaller the edge length, the higher the impact of noise to generate incorrect orientation of the normals. For a number of 800 triangles, more than 5% of the normals have the wrong orientiation when we impose a moderate noise level of 20% noise. For a higher number of 1250 triangles, the number of erroneous orientations rises to more than 8%.

without correction, the widely-used AVV is highly sensitive to noise. With our correction step, AVV becomes robust even in high noise contexts. Higher mesh resolutions are also observed to lead to better results with orientation correction, as expected. Our proposed normals orientation correction results in a clear improvement of principal curvatures estimation with the AVV method, by up to a factor of 10.

References

[1] J. Ahrens, B. Geveci, C. Law, C. Hansen, and C. Johnson. 36-paraview: An end-user tool for large-data visualization. *The visualization handbook*, pages 717–731, 2005.

[2] J. C. Carr, W. R. Fright, and R. K. Beatson. Surface interpolation with radial basis functions for medical imaging. *IEEE transactions on medical imaging*, 16(1):96–107, 1997.

[3] R. Faller. *UCD Biophysics 241: Membrane Biology*. https://LibreTexts.org, 2023.

[4] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics (TOG)*, 23(1):45–63, 2004.

[5] I. K. Jarsch, F. Daste, and J. L. Gallop. Membrane curvature in cell biology: An integration of molecular mechanisms. *The Journal of cell biology*, 214(4):375, 2016.

[6] G. Medioni, M.-S. Lee, and C.-K. Tang. *A computational framework for segmentation and grouping*. Elsevier, 2000.

[7] G. Medioni, C.-K. Tang, and M.-S. Lee. Tensor voting: Theory and applications. In R. Deriche and M.-C. Rousset, editors, *Proceedings of RFIA*, volume 2000, 2000.

[8] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In H.-C. Hege and K. Polthier, editors, *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[9] D. L. Page, Y. Sun, A. F. Koschan, J. Paik, and M. A. Abidi. Normal vector voting: Crease detection and curvature estimation on large, noisy meshes. *Graphical models*, 64(3–4):199–229, 2002.

[10] A. Razdan and M.S. Bae. Curvature estimation scheme for triangle meshes using biquadratic Bézier patches. *Computer-Aided Design*, 37(14):1481–1491, 2005.

[11] M. Salfer, J. F. Collado, W. Baumeister, R. Fernández-Busnadiego, and A. Martínez-Sánchez. Reliable estimation of membrane curvature for cryo-electron tomography. *PLOS Computational Biology*, 16(8):e1007962, 2020.

[12] W. Schroeder, K. Martin, and B. Lorensen. The visualization toolkit, 4th edn. Kitware. *New York*, 2006.

[13] M. Szilvási-Nagy. Face-based estimations of curvatures on triangle meshes. *Journal for Geometry and Graphics*, 12(1):63–73, 2008.

[14] C.-K. Tang and G. Medioni. Robust estimation of curvature information from noisy 3D data for shape description. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 426–433. IEEE, 1999.

[15] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of IEEE International Conference on Computer Vision*, pages 902–907. IEEE, 1995.

[16] W.-S. Tong and C.-K. Tang. Robust estimation of adaptive tensors of curvature by tensor voting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):434–449, 2005.