Fast and exact visibility on digitized shapes and application to saliency-aware normal estimation

Romain Negro[1] and Jacques-Olivier Lachaud[1][0000-0003-4236-2133]

Université Savoie Mont Blanc, CNRS, LAMA, F-73000 Chambéry, France
{romain.negro|jacques-olivier.lachaud}@univ-smb.fr

Abstract. Computing visibility on a geometric object requires heavy computations since it requires to identify pairs of points that are visible to each other, i.e. there is a straight segment joining them that stays in the close vicinity of the object boundary. We propose to exploit a specific representation of digital sets based on lists of integral intervals in order to compute efficiently the complete visibility graph between lattice points of the digital shape. As a quite direct application, we show then how we can use visibility to estimate the normal vector field of a digital shape in an accurate and convergent manner while staying aware of the salient and sharp features of the shape.

Keywords: Visibility · Geometric inference · Digital normal estimation · Digital geometry

1 Introduction

Visibility is a fundamental concept in computational geometry and digital topology, with applications ranging from computer vision (occlusions), geometric modeling (feature detection, geodesics) to computer graphics (path tracing, shadowmap). Visibility within polygons in the plane has been extensively studied in the litterature [9] and is still studied in higher dimensional spaces [20]. The general problem has a high complexity. For instance, the 3d visibility complex between n spheres and occluding spheres has a complexity $O(n^4)$ [6]. Exact 3d visibility between polygons and occluding polygons is often cast into 5d Plücker space [19], which nicely represents 3d lines and their respective positions. The computational cost remains high even with decomposition into coarse cells for speed up (several days of computation for 1M triangles at the time).

To increase efficiency, several authors have cast the visibility problem into a digital space, generally $\mathbb{Z}^3$ or its decomposition into cubical cells. Visibility is simplified as whether there is a connected digital straight line joining a pair of cells without occluding cell(s) (e.g. Soille [21] uses Bresenham segments). Coeurjolly *et al.* [4] use a non-symmetric visibility definition and preimage computations to speed up these algorithms. Chica [2] uses a half-cell erosion of the complementary space and Bresenham lines to decide visibility.

We propose an algorithm that solves the following visibility problem: given a digital set $X \subset \mathbb{Z}^d$, two points p, q of X are *visible* whenever the Euclidean

straight line segment $[p, q]$ is never at chessboard distance (∞-distance) greater or equal to 1 from X. It is equivalent to the *cotangency* of p and q in X, as formalized in [12,13], and can be viewed as an inclusion problem between sets of lattice points in this form. Indeed, cells surounding X can be encoded as lattice points (Khalimsky coding), as well as the cells covering lattice directions. The originality of our method is to encode subsets of lattice points as sets of integer intervals and to reduce all the inclusion problems as intersections and translations of integer intervals. For a given lattice vector $\mathbf{t}$, we solve the visibility of every pair of points $(p, p + \mathbf{t})$ of X. Global visibility is achieved by testing all sought directions within a given range, and is parallelized straightforwardly.

We focus on solving efficiently this problem because it is crucial in computing geometric normals along digital surfaces that are aware of salient features. Indeed, 2d discrete tangent estimators based on tangency or discrete line segments have proven to be both sensitive to sharp corners and multigrid convergent on the boundary of digitized smooth shapes [7,11,18]. Many ad hoc methods have been proposed for tangent plane or normal estimation on 3d digital surfaces (e.g. [8,1,5,10,17]). If two of these methods [5,10] have been shown to be multigrid convergent on digitization of shapes with smooth boundary, their formulation intrinsically smoothes features, e.g. in contrast with the method of Marêché *et al.* [17], whose convergence is not established. We propose here a normal estimator defined as the most orthogonal vector to the cone of visibility at the point of interest. Visibility is high in smooth regions and the induced visibility vectors are good approximations of tangent vectors. On the contrary the visibility is blocked at sharp features, so the normal vector is not influenced by the geometry on the other side of the features. We demonstrate the superiority of this normal estimator for estimating curvatures along digital surfaces with sharp features.

Note that our algorithm for computing visibility is in fact much more general. The same algorithm could be used to recognize if any pattern of lattice points is present on a shape. It is similar to an erosion algorithm where the structuring element is the chosen pattern, and could also be used for morphological operations [22]. The paper outline is as follows. In Section 2, we recall some definitions and present tangency of chords and its equivalence to visibility. Section 3 describes the specific data structure for encoding set of grid cells, and the main algorithm that exploits this encoding to compute visibility exactly. Section 4 presents experimental results on visibility, its usage as a discrete normal estimator, and how it improves curvature estimates on digital surfaces. Finally, Section 5 concludes and gives some perspectives to this work.

2 Visibility through tangency of chords

All the theory and algorithms presented in this section are defined and valid in arbitrary dimension d. They will however be illustrated in 2d for clarity, while experiments will be performed on 3d digital shapes. Let $\mathbb{Z}^d$ be the d-dimensional digital space. Let $\mathcal{C}^d$ be the (cubical) cell complex induced by the lattice $\mathbb{Z}^d$: its 0-cells are the points of $\mathbb{Z}^d$, its 1-cells are the open unit segments joining two

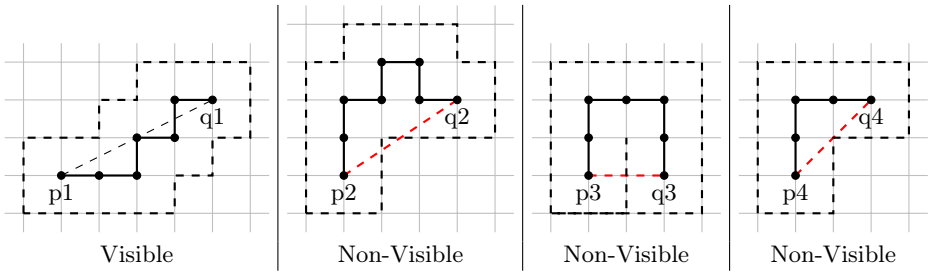

Fig. 1. Examples of visibility and non visibility in 2D within the set X (represented with black dots $\bullet$).

0-cells at distance 1, its 2-cells are the open unit squares, ..., and its d-cells are the d-dimensional open unit hypercubes with vertices in $\mathbb{Z}^d$. We denote by $\mathcal{C}_k^d$ the set of its k-cells, for $0 \leqslant k \leqslant d$. In the following, a cell always designates an element of $\mathcal{C}^d$, and the term subcomplex always indicates a subset of $\mathcal{C}^d$.

The *topological closure* of a cell τ is denoted by $\bar{\tau}$. The *star* of a cell σ is the subcomplex $\mathrm{Star}\,(\sigma) := \{\tau \in \mathcal{C}, \sigma \subset \bar{\tau}\}$. It is naturally extended to any subcomplex K by union, i.e. $\mathrm{Star}\,(K) := \cup_{\sigma \in K} \mathrm{Star}\,(\sigma)$, and more generally to any subset Y of $\mathbb{R}^d$ as $\mathrm{Star}\,(Y) := \{c \in \mathcal{C}^{\mathrm{d}}, \bar{c} \cap Y \neq \emptyset\}$. The *body* $\|K\|$ of K is its realization in $\mathbb{R}^d$, i.e. $\|K\| := \cup_{c \in K} c$. One can check that $\mathrm{Star}\,(K) = \mathrm{Star}\,(\|K\|)$, so these definitions are sound.

Definition 1 (Visibility). *Let $Z \subset \mathbb{Z}^d$ be a digital set. Two digital points p and q are* visible in Z *if and only if the straight segment $[p, q]$ is included in* $\mathrm{Star}\,(Z)$*, i.e.,* $\mathrm{Star}\,([p, q]) \subseteq \mathrm{Star}\,(Z)$.

This definition corresponds to the concept of tangency presented in [13], limited to pairs of points. One can check that p and q must belong to Z to be visible. Figure 1 illustrates what is considered visible or not on some examples.

A quite unexpected fact about visibility it that the set of visible points from a source p is not necessarily digitally connected in X. Indeed, for $n \in \mathbb{Z}, n \geqslant 1$, we say that p, q are *n-neighbors* when $\|q - p\|_\infty \leqslant n$, and this induces the n-adjacency and n-connectedness relation in X. As illustrated on Figure 2, the set of points visible from p is clearly not 1-connected. As our experiments have shown, we even conjecture that it may not be n-connected for n arbitrarily large.

This characteristic has implications for the applicability of [13, Algorithm 3], which is a breadth-first like algorithm for computing all the points of X visible in X to a source point p. This algorithm sometimes outputs an incomplete collection of visible points. This is why in the next section we present an algorithm that computes the exact visibility pairs within X, without recourse to connectedness in its formulation.

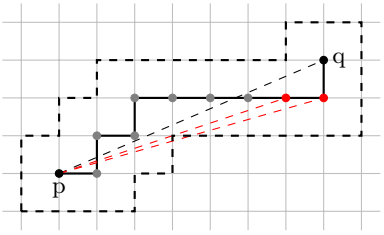

Fig. 2. The set of visible points in X from source p is not 1-connected.

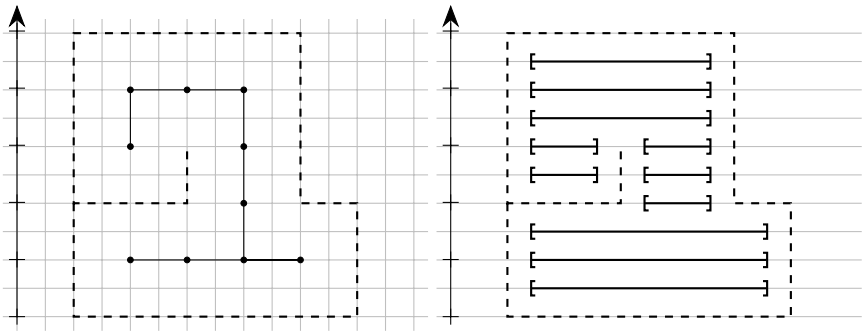

Fig. 3. Representation of a 2d cell complex (here the star of a given curve) as a lattice map. From 63 cells (with two coordinates) on the left, we get 11 intervals on the right.

3 Fast computation using integer intervals intersections

As said in the introduction, any cell of $\mathcal{C}$ can be characterized as a point with integer coordinates, aka *lattice point*. For instance, a standard way is to double the coordinates of the centroid of the cell (often called its Khalimsky code). Now we have to represent an arbitrary set K of lattice points.

Let $(\mathbf{e}_j)_{j=1,\ldots,d}$ be the canonical basis of $\mathbb{Z}^d$. We may choose an arbitrary projection *axis* $j \in \{1,\ldots,d\}$. Letting π_j be the projector along this axis, we can collect all the cells of K that projects onto the same points. Let $p \in K$. Then $\pi_j^{-1}(p) \cap \mathrm{Star}\,(K)$ is a set of cells having all the same coordinates except along coordinate j. This set can be ordered increasingly and stored as a list of integer intervals (see Figure 3 for an illustration).

A *sequence of intervals* is an ordered list of intervals $L = ([a_i, b_i])_{i=1,\ldots,n}$ such that $a_i \in \mathbb{Z}, b_i \in \mathbb{Z}, b_i + 1 < a_{i+1}$, i.e., disjoint intervals with at least a missing integer. We denote by $\mathbb{L}$ the set of sequences of intervals. An integer k belongs to L iff there exists i such that $a_i \leqslant k \leqslant b_i$. For a finite set of integers, there is a unique sequence of intervals representing it.

A *lattice map along axis* j is a set of pairs (q, L_q), with $q \in \mathbb{Z}^{d-1}$ and $L_q \in \mathbb{L}$. The lattice map M_K *represents* K when, for any point $p \in K$, the pair $(\pi_j(p), L_{\pi_j(p)})$ exists in M_K and $p_j \in L_{\pi_j(p)}$, with p_j the j-th coordinate of p,

and reciprocally, $\forall (q, L_q) \in M_K, \forall i \in L_q, q + i\mathbf{e}_j \in K$. A lattice map is thus made of pairs (q, L_q), where q is the *shift* while L_q is the *intervals*. For a point q which is a shift of M_K, we write $M_K[q]$ for the corresponding intervals L_q.

The global algorithm for computing visibility is given in Algorithm 1. Given a digital input surface C (a subcomplex), we compute the lattice map Ω of $\mathrm{Star}\,(C)$ once, generally along the axis where C is the most elongated. In order to compute visibility up to a given distance r, we consider all non-null primitive vectors D with infinity norm no greater than r. For each vector $\mathbf{v}$, we compute the lattice map $M_{\mathbf{v}}$ of its intersected cells (as a vector from $\mathbf{0}$ to $\mathbf{v}$ in $\mathbb{R}^d$). The visibility along the direction $\mathbf{v}$ reduces to computing for each pair of $M_{\mathbf{v}}$ the possible translations of intervals in Ω, and then intersecting all the translations.

Algorithm 1 Given a subcomplex C and an integer r, returns the visibility from every point of C up to distance r. The main axis is supposed to be z, while x, y are the auxiliary axes.

```
function Visibility(C: Subcomplex, r: Integer)
    Ω ← M_Star(C)                          ▷ Lattice map of the star of input subcomplex
    Directions ← GetAllPrimalDirections(r)
    V : vector of boolean ← [0, ..., 0]    ▷ length Size(Directions) × #C.pointels
    low, high ← BoundingBoxZ(Ω)
    for all v in Directions do
        for all shift S in Ω do
            R ← [low, high]
            for all pair P in M_Star([0,v]) do
                R ← R ∩ Translations(P.intervals, Ω[S + P.shift])
            UpdateVisibility(V, R)
    return V
```

Figure 4 shows an example of execution of this algorithm in an elementary case. It reduces to determining all the possible inclusions by translation of sequences of intervals in another sequence of intervals (note that for pairwise visibility, it is enough to consider all the possible inclusions by translation of one interval in another sequence of intervals), and then intersecting progressively the results. Of course, the process is stopped as soon as the intersection is empty. Some results of visible points are displayed on figure 5 on 3d shapes.

To efficiently compute the intersection of two sequences of intervals K and L (Algorithm 2), we go through the first 2 unvisited elements of the 2 lists. If they do not overlap (i.e. the start and end of the 1st interval are both smaller than the start of the other), then we can discard the smaller interval. Else, we construct the intersection of these 2 intervals as one of the resulting intervals. We then can skip the interval with the smallest end. If both intervals have the same end, then both intervals can be skipped. Doing this computation until one of the lists is empty returns the intersection of both lists of intervals.

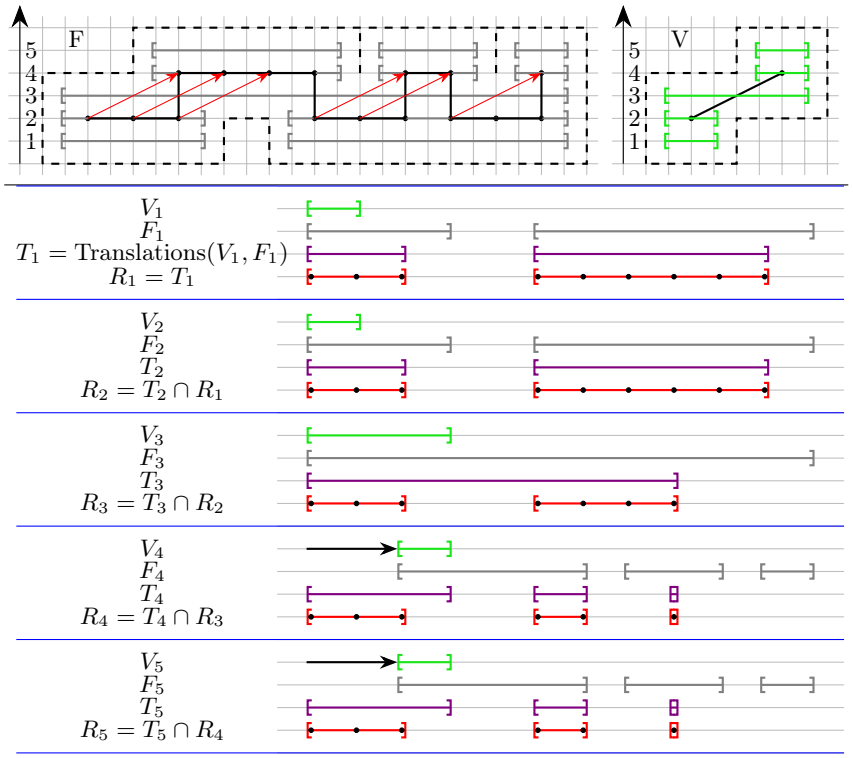

Fig. 4. Evolution of the visibility check algorithm for a $(2, 1)$ vector. Green is the vector lattice map, black is the figure lattice map, red are the current intervals of positions where the visibility is still possible. The last red intervals are the visible positions. We travel the lattice maps from bottom-up and the found visibilities are drawn on the uppest figure.

Algorithm 2 Given 2 lists of integer intervals K and L, returns $K \cap L$

function INTERSECTION $(\cap)$(K, L: Intervals)
  $R \leftarrow \emptyset$; $k \leftarrow 0$; $l \leftarrow 0$;
  while $k < K.nbIntervals$ & $l < L.nbIntervals$ **do**
    $[a, b] \leftarrow K[k]$; $[c, d] \leftarrow L[l]$
    $e \leftarrow \max(a, c)$; $f \leftarrow \min(b, d)$;
    if $e \leq f$ **then** $R.append([e, f])$;
    if $b \leq d$ **then** $k \leftarrow k + 1$;
    if $d \leq b$ **then** $l \leftarrow l + 1$;
  return R

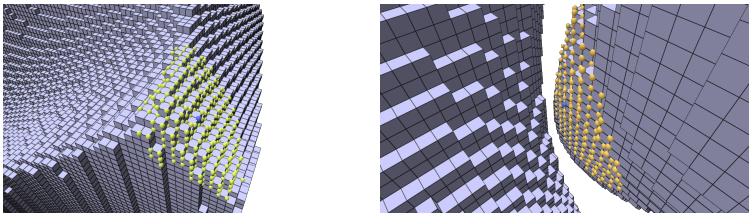

Fig. 5. Examples of visibilities from a source point: (left) the visibility is stopped at the sharp edge, (right) the visibility does not cross the gap between the two parts.

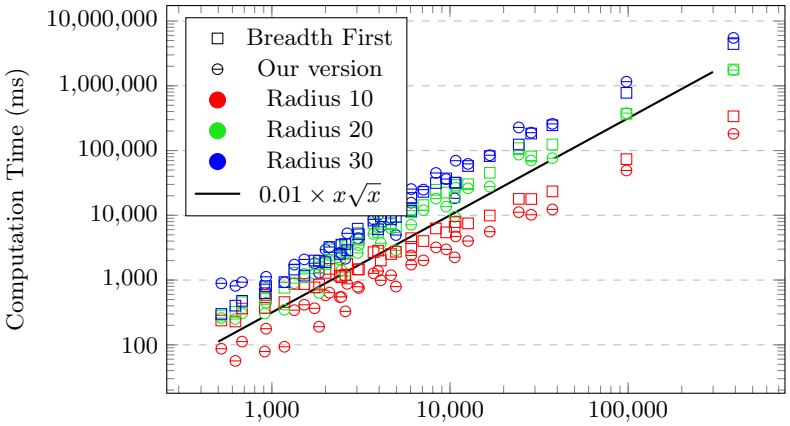

Fig. 6. Computation time of visibility as a function of the number of pointels. We compare the running time of the naive breadth first visibility algorithm against our algorithm using intervals. We input the same maximal radius to both versions (10, 20, 30) and test it on 6 different figures (goursat, torus, rcube, sphere9, leopold, "D20") with various gridsteps, (# pointels ranging from 520 to 390235).

We have plotted on Figure 6 the running times to compute global visibility of both [13, Algorithm 3] and our algorithm. Timings are very similar. Our algorithm is faster for smaller maximal radii (up to 20), while the other is slightly faster for bigger radii. We have also measured the (discrete) average visibility distance along digitization of smooth shapes (Figure 7). It follows a law proportionnal to $\sqrt{h}$. As shown in the next section, for real 3d images, a maximal radius below 10 is sufficient for normal and curvature estimation.

4 Saliency-aware normal and curvature estimation

Saliency-aware normal estimator. We propose a new normal estimator on digital surfaces, which uses the visibility of the point of interest while taking into account a user-given scale σ. If σ is proportionnal to the average distance between visible points (so some $\Theta\left(\sqrt{h}\right)$), then this estimator is observed to be multigrid

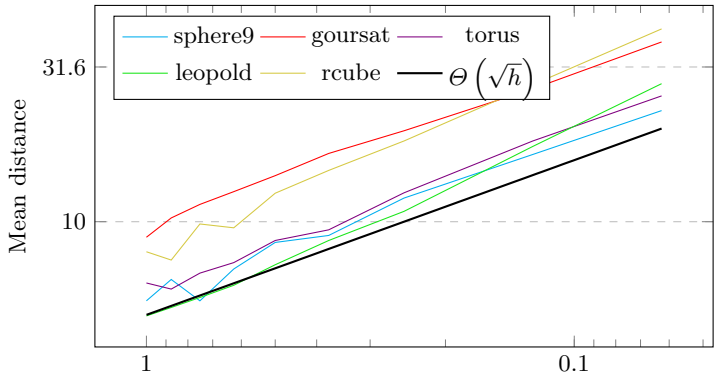

Fig. 7. Mean distance of visibility as a function of the gridstep h (see text).

convergent along digitization of smooth shapes. However, since the computation window is limited by the visibility, it better approximates the geometry near sharp or salient features.

More precisely, we choose a Gaussian function $w_\sigma(x) \coloneqq e^{-\frac{x^2}{2\sigma^2}}$ as a weight kernel. Let V_p be the set of visible points from the point p. Let c_p be the weighted centroid of the visible points around p, i.e. $c_p \coloneqq \frac{\sum_{q\in V_p} w_\sigma(\|q-p\|)q}{\sum_{q\in V_p} w_\sigma(\|q-p\|)}$. We form the weighted covariance matrix $\mathcal{V}_p$ of the points V_p as:

$$\mathcal{V}_p = \sum_{q\in V_p} w_\sigma(\|q-p\|)(q-c_p)(q-c_p)^T. \tag{1}$$

The *visibility normal* $\boldsymbol{n}(p)$ of point p at scale σ is defined as the first eigenvector of the covariance matrix $\mathcal{V}_p$ of the visible points, corresponding to its smallest eigenvalue. Its orientation is chosen to point in the same direction as the average of the trivial normals to the surfels touching the point p.

Experimental validation. Figure 9 shows an example of our normals compared to normals computed with the integral invariant (II) estimator [10]. In order to compare quantitatively the different normal estimators, we compute the root-mean-square error (RMSE) and the maximum error $E_{\max}$ of the normals computed on a digital surface with respect to the normals computed on the continuous surface (Figure 8). We use 4 different digital surfaces (ellipsoid, goursat, rcube and sphere9). We see that the RMSE and $E_{\max}$ errors of our estimator are convergent with respect to the grid resolution with a rate of convergence of $h^{\frac{2}{3}}$ for the RMSE and $h^{\frac{1}{2}}$ for $E_{\max}$.

Application to curvature estimation. In order to illustrate better that the visibility normals better take into account the sharp features of a digital surface while staying meaningfull in digitization of smooth parts, we compare the curvature estimates induced by different discrete normal estimators. We exploit the

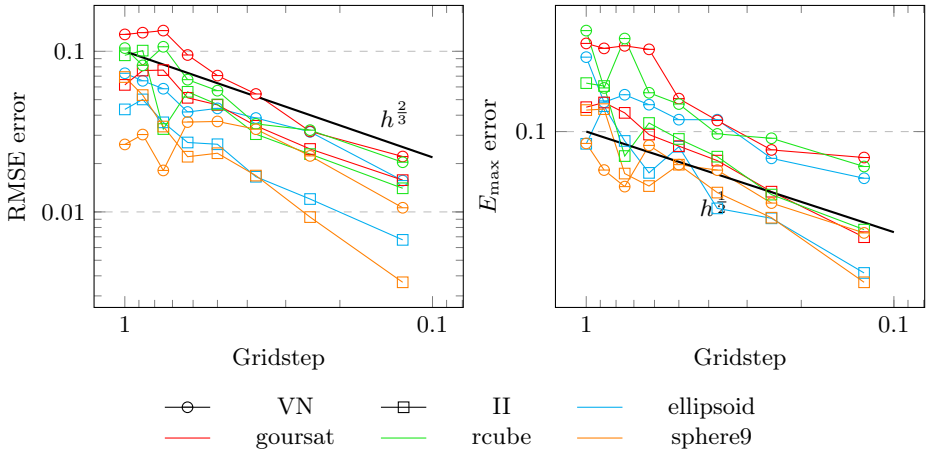

Fig. 8. Error RMSE and $E_{\max}$ as a function of grid resolution

Corrected Normal Current (CNC) estimator, whose theory is detailed in [15] and which can estimate all kinds of curvatures given positions and normals. It produces state-of-the-art curvature estimates on polygonal surfaces [16], point clouds [14], and even outperforms Integral Invariant curvature estimates [3] on digital surfaces [15]. Its implementation is also available in DGtal.

Figure 10 displays the results of CNC curvature estimations on the digitization of a piecewise smooth shape "Talking D20", taken from Thingi10K. We compare the differences obtained by just changing the discrete normal estimator (available in the DGtal library): (column II normals) Integral Invariant normals with $r = 6$, (column CTriv normals) convolved trivial normals with $r = 6$, (column Visibility normals) our proposed estimator (VN) with $\sigma = 4$ and maximal visibility distance 2σ. These parameters were chosen so that the respective computation windows of the different estimators are approximately the same.

The II estimator is good along digitization of smooth or flat parts of the dice, but presents curious artefacts near sharp features induced by the hollowed out numbers: this is due to the nature of II normal estimates, which computes a PCA of a ball centered on the point of interest and may include points on the other side of the saliencies.

The CTriv estimator only performs averaging of trivial normal directions along the surface. It behaves much better than II near features (although it tends to smooth curvatures) since it does not use information from across the gap. However, some oscillations of the curvatures are distinguishable especially along the flat parts, and some curvature estimates are erroneous on smoother parts (like the edge above 6 for κ_2 or several dice vertices).

The VN estimator takes the best of the two previous approaches. It remains precise and stable on smooth and flat regions, while perfectly delineating sharp features and holes, whatever the kind of estimated curvatures. Its drawback is

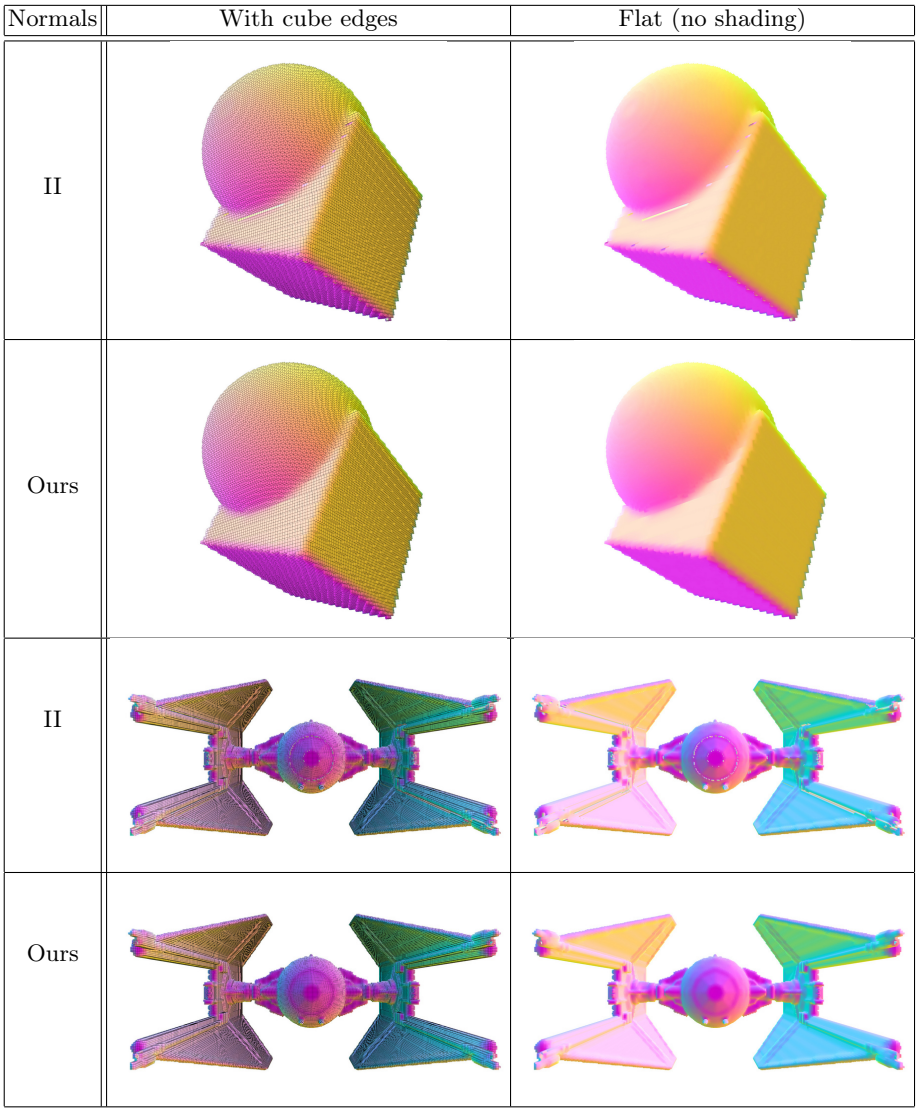

Fig. 9. Examples of normals computed on a cps, then on a tie as a color map. First set displays II normals, second set uses the visibility algorithm to compute them. The normals are colored according to their orientation as $0.5 \cdot (n_i + \mathbb{1})$. The right images are the smooth version of the left images. "Integral Invariant normals" are computed using a radius of $r = 4.5$, while our normal estimator are computed using a deviation of $\sigma = 4$, and so a radius of $r = 2\sigma = 8$

the computation time, which is $52s$ for a maximal visibility distance 8, compared to 2.3s for II and $0.1s$ for CTriv. Note that the computation time falls back to $33s$ for a maximal distance of 6 (and same $\sigma = 4$) while the result is visually indistinguishable.

5 Conclusion

We have proposed a new algorithm that computes the visibility between all pairs of points on a digital surface. It uses the specificity of a structure to represent sets of lattice points or cells, a mapping from projected coordinates to list of integer intervals. Compared to a classical breadth-first approach to compute visibility, it provides the exact visibility result, even when the set of visible points from a source is disconnected. As shown by experiments the running times are comparable with the breadth-first algorithm, and even faster for small maximal visibility distance, which are typical when processing standard 3d images up to 512^3. We have used visibility to define a discrete tangent estimator that respects salient features while giving good approximations on digitization of smooth parts. The obtained normal field is more suitable to curvature estimation than the classical Integral Invariant normal estimator.

We have several ideas for possible future works. First, it is possible to speed up the identification of non visibility by a coarse-to-fine approach and a precomputation of a pyramid of covering cells. We would like to explore if we can build such a pyramid variant of our algorithm, considering the primitive vectors as a hierarchy. Second we would like to prove the muligrid convergence of our normal estimator, which is observable on experiments. Last, our algorithm is not limited to pairwise visibility, but is more an algorithm of exact pattern matching. We would like to explore new applications of pattern identification along digital surfaces (like corner detection, locally convex zones identification, etc.).

Acknowledgments. This work is partially supported by the French National Research Agency within the StableProxies project (ANR-22-CE46-0006).

References

1. Charrier, E., Lachaud, J.O.: Maximal planes and multiscale tangential cover of 3d digital objects. In: Proc. Int. Workshop Combinatorial Image Analysis (IWCIA2011). LNCS, vol. 6636, pp. 132–143. Springer Berlin / Heidelberg (2011)
2. Chica, A.: Visibility-based feature extraction from discrete models. In: Proceedings of the 2008 ACM symposium on Solid and physical modeling. pp. 347–352 (2008)
3. Coeurjolly, D., Lachaud, J.O., Levallois, J.: Multigrid convergent principal curvature estimators in digital geometry. Computer Vision and Image Understanding **129**, 27–41 (2014)
4. Coeurjolly, D., Miguet, S., Tougne, L.: 2d and 3d visibility in discrete geometry: an application to discrete geodesic paths. Pattern Recognition Letters **25**(5), 561–570 (2004)

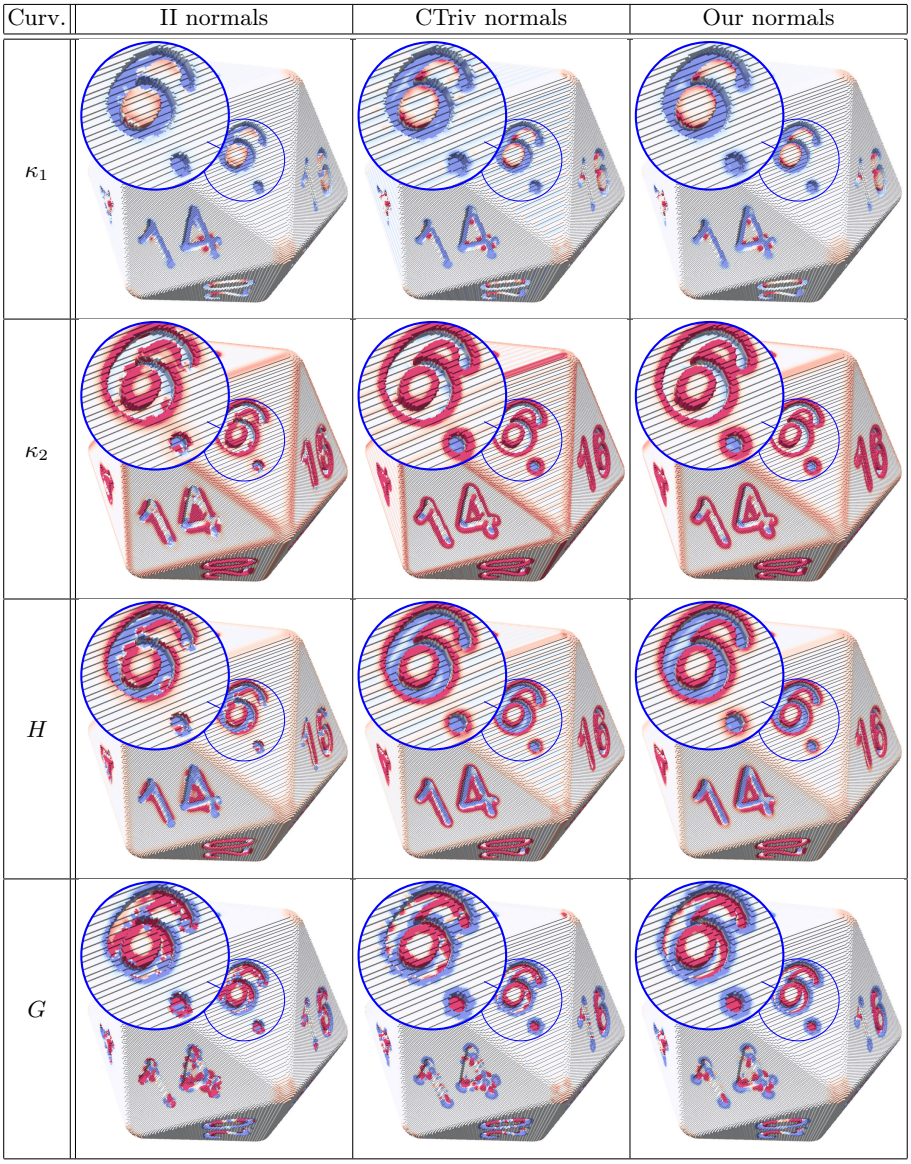

Fig. 10. Estimation of curvatures using Corrected Normal Current estimators [15] with a measure of radius 2 for the shape "Dice-20" of Thingi10K database at resolution 256^3. This estimator is parameterized by an input normal estimator: first column by "Integral Invariant normals" (radius $r = 6$), second column by "Convolved Trivial normals" (radius $k = 6$), third column by our presented normal estimator (deviation $\sigma = 4$, radius $r = 2\sigma = 8$). Per row are displayed the estimated curvatures in order: first and second principal curvatures κ_1 and κ_2, mean curvature H, Gaussian curvature G. The range of curvature between deep blue and deep red is $[-0.1, 0.1]$ with white color equal to 0.

5. Cuel, L., Lachaud, J.O., Thibert, B.: Voronoi-based geometry estimator for 3d digital surfaces. In: Barcucci, E., Frosini, A., Rinaldi, S. (eds.) Proc. Int. Conf. on Discrete Geometry for Computer Imagery (DGCI'2014), Sienna, Italy. LNCS, vol. 8668, pp. 134–149. Springer International Publishing (2014). https://doi.org/10.1007/978-3-319-09955-2_12
6. Durand, F., Drettakis, G., Puech, C.: The 3d visibility complex. ACM Transactions on Graphics (TOG) **21**(2), 176–206 (2002)
7. Feschet, F., Tougne, L.: Optimal time computation of the tangent of a discrete curve: Application to the curvature. In: Discrete Geometry for Computer Imagery: 8th International Conference, DGCI'99 Marne-la-Vallée, France, March 17–19, 1999 Proceedings 8. pp. 31–40. Springer (1999)
8. Fourey, S., Malgouyres, R.: Normals estimation for digital surfaces based on convolutions. Computers & Graphics **33**(1), 2–10 (2009)
9. Ghosh, S.K.: Visibility algorithms in the plane. Cambridge university press (2007)
10. Lachaud, J.O., Coeurjolly, D., Levallois, J.: Robust and convergent curvature and normal estimators with digital integral invariants. In: Najman, L., Romon, P. (eds.) Modern Approaches to Discrete Curvature, Lecture Notes in Mathematics, vol. 2184, pp. 293–348. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-58002-9_9
11. Lachaud, J.O., Vialard, A., de Vieilleville, F.: Fast, accurate and convergent tangent estimation on digital contours. Image and Vision Computing **25**(10), 1572–1587 (October 2007)
12. Lachaud, J.O.: An alternative definition for digital convexity. In: Lindblad, J., Malmberg, F., Sladoje, N. (eds.) Discrete Geometry and Mathematical Morphology - First International Joint Conference, DGMM 2021, Uppsala, Sweden, May 24-27, 2021, Proceedings. LNCS, vol. 12708, pp. 269–282. Springer (2021). https://doi.org/10.1007/978-3-030-76657-3_19
13. Lachaud, J.O.: An alternative definition for digital convexity. J. Math. Imaging Vis. **64**(7), 718–735 (2022). https://doi.org/10.1007/s10851-022-01076-0
14. Lachaud, J.O., Coeurjolly, D., Labart, C., and, P.R.: Lightweight curvature estimation on point clouds with randomized corrected curvature measures. Computer Graphics Forum **42**(5), e14910 (2023)
15. Lachaud, J.O., Romon, P., Thibert, B.: Corrected curvature measures. Discret. Comput. Geom. **68**(2), 477–524 (2022). https://doi.org/10.1007/s00454-022-00399-4
16. Lachaud, J.O., Romon, P., Thibert, B., Coeurjolly, D.: Interpolated corrected curvature measures for polygonal surfaces. Computer Graphics Forum **39**(5), 41–54 (2020). https://doi.org/10.1111/cgf.14067
17. Marêché, A., Debled-Rennesson, I., Feschet, F., Ngo, P.: A parameter-free normal estimator on digital surfaces. In: International Conference on Intelligent Systems and Pattern Recognition. pp. 130–142. Springer (2024)
18. Nguyen, T.P., Debled-Rennesson, I.: A discrete geometry approach for dominant point detection. Pattern Recognition **44**(1), 32–44 (2011)
19. Nirenstein, S., Blake, E., Gain, J.: Exact from-region visibility culling. In: Proceedings of 13th Eurographics Workshop on Rendering (2002)
20. O'Rourke, J.: Visibility. In: Handbook of discrete and computational geometry, pp. 875–896. Chapman and Hall/CRC (2017)
21. Soille, P.: Generalized geodesy via geodesic time. Pattern Recognition Letters **15**(12), 1235–1240 (1994)
22. Soille, P., et al.: Morphological image analysis: principles and applications, vol. 2. Springer (1999)