Visibility in Discrete Geometry: an application to discrete geodesic paths

D. Coeurjolly, S. Miguet and L. Tougne [a]

[a]Laboratoire ERIC
Université Lumière Lyon 2
5, av. Pierre Mendès-France
69676 BRON cedex, FRANCE

In this article, we present a discrete definition of the classical visibility in computational geometry based on digital straight lines. We present efficient algorithms to compute the set of pixels in a non-convex domain that are visible from a source pixel. Based on these definitions, we define discrete geodesic paths in discrete domain with obstacles. This allows us to introduce a new geodesic metric in discrete geometry.

1. Introduction

In discrete geometry, many Euclidean geometric tools are redefined to take into account specificities of the discrete grid. In this article, we propose a definition of the classical Euclidean visibility based on discrete objects. The interest is double: on one hand we extend the discrete geometry with a new tool and on the other hand, since this visibility allows us to define discrete geodesic paths and discrete shortest paths, we have a practical tool needed by many applications in medical imaging or image analysis to estimate geodesic distance in non-convex domains.

In the literature, [25–27] defines the visibility between two points using the Bresenham digital line drawing algorithm [4]. The visibility definition we propose is based on classical Discrete Straight Lines (DSL for short). Hence the proposed approach consider a more general set of lines and allow efficient algorithms. Many technics exist for the DSL recognition problem. Some of these approaches are based on chain code analysis [31], on links between the chain code and arithmetical properties of DSL [8,9], on links between the chain code and the feasible region in the dual -or parameter- space [11,19,30] and others on linear programming tools such that Fourier-Motzkin's algorithm [13]. All these algorithms present a solution either to decide if a given set of pixels is a discrete straight segment (DSS for short) or to segment a discrete curve into DSS, or both. In our case, the problem is quite different, we want to decide if there exits a DSS between two pixels in a non-convex domain.

We present definitions and algorithms to compute the set of pixels which are visible from a source. Then, we define a notion of discrete geodesic path and a metric associated to such path based on this visibility definition. We also proposed an efficient implementation of the geodesic distance labelling from a source pixel.

2. Visibility

2.1. Notions and definitions

Let us denote $\mathcal{D}$ a *discrete domain*, that is a $n-$connected set of pixels. We denote $\bar{\mathcal{D}}$ the complement of $\mathcal{D}$, we call this set indifferently the *background* or the *set of obstacles*. In the following, we consider $\mathcal{D}$ a 8-connected domain.

In this domain, we define the discrete visibility by analogy to the continuous definition.

Definition 1 (Discrete Visibility) *Let s and t be two pixels in $\mathcal{D}$, we define the discrete visibility as a binary relationship $v : \mathcal{D} \to \mathcal{D}$ such that we have $v(s,t)$ if and only if there exists a 8-connected discrete straight segment from s to t whose pixels belong to $\mathcal{D}$*

Before introducing the visibility problem in non-convex domain, we recall classical parameter space characterizations of DSL [19,20,30]. If we consider an Euclidean straight line $y = \alpha x + \beta$, the digitization of this line using the Grid Intersect Quantization (see [14] for a survey on digitization scheme) is the set of discrete points such that:

$$\Delta(\alpha,\beta) = \{(x,y) \in \mathbb{Z}^2 | -\frac{1}{2} \leq \alpha x + \beta - y < \frac{1}{2}\}$$

Note that all classical digitization schemes (GIQ, Object Boundary Quantization or Background Boundary Quantization) can be used and such a choice will not interfere in our algorithms. We choose the GIQ scheme because of its symmetry properties.

In the parameter space of the previous definition, we can describe the set of Euclidean straight lines whose the digitization contains a pixel $p(x,y)$:

$$S_p = \{(\alpha,\beta) \in \mathbb{R}^2 | -\frac{1}{2} + y \leq \alpha x + \beta < \frac{1}{2} + y\}$$

A pixel in $\mathcal{D}$ defines a *strip* in the (α,β)-space delimited by two lines $L_1 : \alpha x + \beta - y \geq -\frac{1}{2}$ and $L_2 : \alpha x + \beta - y < \frac{1}{2}$. If we want to know if a set of pixels belongs to a DSL, a classical way is to compute the intersection in the (α,β)-space of strips associated to each pixel. If the feasible domain is not empty, it describes all DSL containing the pixels (cf figure 2 for an example). In the following, we define the domain $\mathcal{S}(s,t)$ associated to pixels s and t the, intersection $S_s \cap S_t$.

In order to compute the visibility in non-convex domains, the main idea is to check in the dual space if domains associated to obstacle pixels do not hide the current pixel t from the source s.

2.2. Visibility domain

Let o denote an obstacle pixel. If we want to describe the set of Euclidean straight lines whose digitizations do not contain o, we also introduce a strip in the parameter space such that the inequations are reversed. Hence, an obstacle o is associated to constraints $\bar{L}_1(o) : \alpha x + \beta - y < -1/2$ and $\bar{L}_2(o) : \alpha x + \beta - y \geq 1/2$. If we want to know if this obstacle *blocks* the visibility from s to t, we just have to compute in the (α,β)-space $L_1(s) \cap L_2(s) \cap L_1(t) \cap L_2(t) \cap \bar{L}_1(o) \cap \bar{L}_2(o)$. If this intersection is empty then t is not visible from s.

More generally, if we consider a non-convex domain $\mathcal{D}$ and a set of obstacle pixels $\mathcal{O} = \{o_i\}_{i=1..n}$ that is a restriction of $\bar{\mathcal{D}}$ such that all point abscissas are between the abscissa of s and the abscissa of t (all other points can be omitted for the visibility problem). We have the lemma:

Lemma 1 *Let s be the source and t a pixel in $\mathcal{D}$, t is visible from s in $\mathcal{D}$ if and only if:*

$$\mathcal{S}(s,t) \cap \left(\bigcap_{i=1..n} \bar{L}_1(o_i) \cap \bar{L}_2(o_i) \right) \neq \emptyset$$

The proof of this lemma can be deduced by the visibility definition and by construction of $\mathcal{S}$.

Obviously, we do not have to consider all obstacle pixels. We first define:

Definition 2 *A pixel o in $\mathcal{O}$ is called "blocking pixel" for the visibility problem $v(s,t)$ if:*

$$\mathcal{S}(s,t) \cap \bar{L}_1(o) \cap \bar{L}_2(o) \neq \mathcal{S}(s,t)$$

and the abscissa of o is between the abscissa of s and t.

These blocking pixels are those which interfere in the visibility problem. Non-blocking pixels in $\mathcal{O}$ can be removed from the $v(s,t)$ test. We can characterize the shape of the domain when a blocking pixel modifies it:

Lemma 2 *If o is a blocking pixel for the $v(s,t)$ problem, either the domain $\mathcal{S}(s,t) \cap \bar{L}_1(o) \cap \bar{L}_2(o)$ is empty or it has only one connected component.*

Proof: we consider the domain $\mathcal{S}(s,t)$ and a blocking pixel o such that o, s and t are not collinear (in that case, the domain is empty). We show that either $\bar{L}_1(o)$ or $\bar{L}_2(o)$ crosses the domain. We have different cases (cf figure 1-a) that induce two components but the left and the middle cases are excluded because they imply that the abscissa of o denoted x_o is not between x_s and x_t and thus, o is not a blocking pixel according to definition 2. As the matter of fact, if x_o is between x_s and x_t, then the slope of $\bar{L}_1(o)$ is between the slope of $L_1(s)$ and the slope of $L_1(t)$. By construction of the strips, the vertical distance between L_1 and L_2 is equal to 1. Hence, in figure 1-b, the intersection in a' of $\bar{L}_1$ with the vertical line going through b implies that b' must be outside the interval $[a,b]$ on the vertical line. Since the slope of $\bar{L}_2$ is greater than the slope of the edge cb, $\bar{L}_2$ cannot cross the domain. Same idea can be applied when $\bar{L}_2$ crosses the domain. Hence, all cases of the figure 1-a are impossible and thus, $\mathcal{S}(s,t)) \cap \bar{L}_1(o) \cap \bar{L}_2(o)$ has only one connected component. □

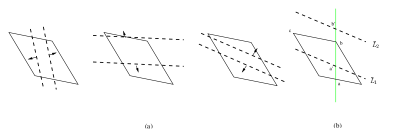

Figure 1. a) Different cases that induce two connected components, the left case and the middle case are impossible by definition of blocking points. The third case must be taken into account b) illustratcfion of the proof of lemma 2.

According to this lemma, if a straight line $\bar{L}_1$ (resp. $\bar{L}_2$) of an obstacle crosses the domain, the other constraint $\bar{L}_2$ (resp. $\bar{L}_1$) can be removed for the visibility problem. Geometrically, an obstacle such that $\bar{L}_1$ crosses the domain is above the Euclidean segment $[s,t]$ and an obstacle such that $\bar{L}_2$ crosses the $\mathcal{S}(s,t)$ domain is beneath the segment $[s,t]$ (cf figure 2 for an example). We denote $\mathcal{U}(s,t)$ the set of blocking pixels above $[s,t]$ and $\mathcal{L}(s,t)$ the set of blocking pixels beneath the segment $[s,t]$.

In [25–27], the visibility test is computed considering the digitization, using the Bresenham's algorithm [4], of the Euclidean segment $[st]$ and verifying that all pixels of this segment belong to the domain. In our proposal, we consider all possible digital straight segment and thus we increase the visibility domain. Beyond these different definitions, we present efficient algorithms for both visibility labelling and geodesic distance labelling.

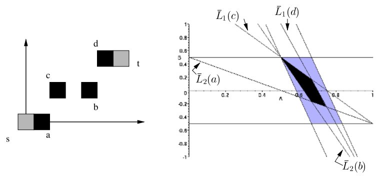

Figure 2. Visiblity domain associated to a set of blocking pixels. The black feasible region in the parameter space is the visibility domain associated to grey pixels constrained with the black blocking pixels.

2.3. Visibility algorithm

In this section, we present algorithms that compute the equivalence class associated to the visibility binary relationship of a source s.

We propose two algorithms, the first one computes the equivalence class with the visibility definition given above, and the second one introduces a new visibility definition that is a restriction of the previous one but the associated algorithm complexity justifies this new version of the visibility.

The first algorithm we propose is a really straightforward computation of the visibility. Indeed, we can use classical linear programming tools to solve the linear inequation system given by obstacle constraints. Such tools are for example the Fourier-Motzkin [13] system simplification algorithm, the Simplex algorithm or the Megiddo's algorithm [21]. Note that the complexity of the Megiddo's algorithm is linear in the number of inequations but the problem comes with the dimension of the system. In our case, the constraint system is in dimension 2 and thus the implementation of the Megiddo's algorithm is tractable with a complexity bounded by $4n$ where n is the number of inequations.

We consider a source s, a domain $\mathcal{D}$. We label all pixels in $\mathcal{D}$ using a breadth-first tracking of the domain using for example the 8-adjacency.

During the propagation process, if we meet an obstacle we store its coordinates in a list O. At each pixel visited in the breadth-first tracking, we extract from O the set of blocking pixels and we solve the visibility problem using the Megiddo's algorithm.

Straightforward visibility algorithm
Input: a domain $\mathcal{D}$ and a source s
Output: the set of pixels which satisfy $v(s,t)$

Let Q be a FIFO queue
Let O be the obstacle list
Append_last(s,Q)
While Q is not empty
  t:=remove_first(Q)
  For each 8-neighbor n of t not labelled `closed` or `visible`
    If n is an obstacle **then**
      Append(n,O)
    else
      Let B be the set of blocking points of O according to the pixel n
      Compute the linear inequation system S with $\bar{L}_1$ or $\bar{L}_2$ the constraints of each point of B
      If $Megiddo(S) \neq \emptyset$ **then**
        Label n as `visible`
        Append_last(n,Q)
      else
        Label n as `closed` //n is not visible and the point is closed
  endFor
endWhile

If we denote n the number of pixels in $\mathcal{D}$ and m the number of obstacles in O, each step in the while loop has got a complexity bounded by $O(m)$. Hence, the global cost of this algorithm is $O(nm)$.

Due to the difficulties to provide an efficient data structure to propagate blocking points from a point to its neighbors, this algorithm has a quite important complexity and is not efficient for the geodesic computation. Thus, we propose a new definition of the discrete visibility which is a weak version of the definition presented above but that leads to an efficient algorithm for the visibility computation and the discrete geodesic problem.

Definition 3 (Weak Discrete Visibility) *Let s and t two pixels in $\mathcal{D}$, we define the weak discrete visibility as a binary relationship $v^* : \mathcal{D} \to \mathcal{D}$ such that we have $v^*(s,t)$ if and only if there exists an Euclidean straight line going through s and whose digitization contains t and no pixels in $\bar{\mathcal{D}}$ between s and t.*

Instead of considering the inequation associated to s, we constraint the set of Euclidean lines to go through s. This new definition restricts the previous one and make the visibility not be a symmetric binary relationship. However, this definition allows an efficient data structure for the visibility test. We suppose that all obstacle pixels are sorted by polar angles using s as the origin. Using this data structure and the above definition, we have the following property.

Proposition 1 *Given a set of obstacles sorted by polar angles of center s and a point t, we denote u the minimum of $\mathcal{U}(s,t)$ and l the maximum of $\mathcal{L}(s,t)$. We have:*

$$v^*(s,t) \Leftrightarrow \mathcal{S}^*(s,t) \cap \bar{L}_1(u) \cap \bar{L}_2(l) \neq \emptyset$$

where $\mathcal{S}^*$ denotes the new domain associated to the weak visibility which is now a segment in the parameter space.

Hence, instead of considering all blocking pixels, we just have to test two characteristic pixels given by a polar sort. The proof of this property is a straightforward application of the visibility definition. Note that the polar sort can be done with integer arithmetic.

We can present the algorithm associated to this definition:

Weak visibility algorithm
Input: a domain $\mathcal{D}$ and a source s
Output: the set of pixels which satisfy $v(s,t)$

Let Q be a FIFO queue
Let O be the obstacle list sorted in a polar trigonometric order of center s
Append_last(s,Q)
While Q is not empty
  t:=remove_first(Q)
  For each 8-neighbor n of t not labelled `closed` or `visible`

```
        If n is an obstacle then
            Append_sort(n,O)
        else
            Let (u,l) be the localization of n in the sorted set O
            If S*(s,t) ∩ L̄1(u) ∩ L̄2(l) ≠ ∅ then
                Label n as visible
                Append_last(n,Q)
            else
                Label n as closed //n is not visible and the point is closed
    endFor
endWhile
```

The visibility test has got a constant time cost and according to the data structure, both localization and obstacle insertion have a cost in $O(log(m))$. Thus, the global cost of this algorithm is $O(nlog(m))$. Moreover, the cone (s,u,l) associated to a point t can be propagated for both localization and insertion to reduce the expected complexity of the algorithm that makes this labelling very efficient.

3. Discrete shortest path and discrete geodesic metric

Based on these definitions of the visibility, we can define discrete shortest paths and discrete geodesic paths.

3.1. Definition and previous works

We first remind some classical facts on discrete metrics that approximate the Euclidean one. All discrete metrics are based on:

- either a mask approach where elementary steps in the neighborhood graph are weighted in order to approximate the Euclidean distance of these steps. For example, elementary steps of the Manhattan distance (or d_4) are horizontal or vertical moves weighted to 1, the chess-board distance (or d_8) also considers diagonal moves weighted to 1. More generally, chamfer metrics first list elementary moves and then associate weights to each move (see [2,29] for initial works) ;
- or a vector approach that leads to exact Euclidean metric where displacement vector (dx, dy) is stored and then the distance can be exactly computed $d = \sqrt{dx^2 + dy^2}$ but the main goal is to design distance map algorithm that only deal with the integer displacements [7,24,6].

For the discrete geodesic problem, the mask based approach leads to efficient algorithms because a weighted graph can be computed from the metric and the adjacency graph of the domain $\mathcal{D}$ and thus, classical shortest path algorithms can be applied such as the Dijkstra's graph search algorithm [23].

In the following, we use the data structure and the implementation of the geodesic mask given by [28]. The authors describe an *bucket sorting* implementation of the Dijkstra's graph search algorithm which leads to a uniform cost algorithm.

[6] proposes a region growing Euclidean distance transform using the same structures but the bucket are indexed by the square distance $dx^2 + dy^2$. For all the visible pixels from the source, this algorithm provides a good estimation of the Euclidean distance metric. This algorithm is not error-free but we will discuss this point later.

[22,3] present an algorithm for the geodesic metric problem based on a discrete arc chain code propagation scheme but some operations to maintain the data structure are expensive. In our case, we use a uniform cost data structure from which we can extract arc chain code but the visibility property is propagated instead of iso-metric points.

3.2. Algorithm

The main idea of our discrete geodesic algorithm is the following: for all pixels which are visible from the source, we do not have any problem to compute their distance because it exists a discrete straight line between the source and these points and thus, we can compute the displacement vector and return $\sqrt{dx^2 + dy^2}$. If a pixel p is not visible, we start a new visibility computation such that p is a new source and each pixel t such that $v(p,t)$ will be labelled by the distance from p to the source plus the distance between p and t.

More formally, we have the following purely discrete definition of a geodesic path in $\mathcal{D}$:

Definition 4 (Discrete Geodesic Path) *A discrete geodesic path between a point t and a source s is a sequence of pixels in $\mathcal{D}$ denoted $\{p_i\}_{i=0..n+1}$ with $p_0 = s$ and $p_{n+1} = t$ such that:*

$$v(p_i, p_{i+k}) \quad \textit{iff} \quad k = \{-1, 0, 1\} \quad \textit{with } i = 1..n$$

And such that the geodesic distance $d_{geodes}(s,t)$ is minimal. The geodesic distance is defined by:

$$d_{geodes}(s,t) = \sum_{i=0}^{n} d_{euc}(p_i, p_{i+1})$$

where $d_{euc}(a,b)$ denotes the Euclidean distance between pixels a and b.

The discrete geodesic path is thus a 8-connected curve that is segmented into DSS by construction. The metric we associate to this curve have been intensively studied and both the stability and multigrid convergence have been proved [18,17,5].

In order to design an efficient algorithm based on the Verwer's bucket structure [28], we consider rounded geodesic distance to index the buckets: a pixel p belongs to the bucket d if and only if:

$$\lceil d_{geodes}(s,p) \rceil = d$$

This estimated metric is still consistent for the Verwer's algorithm (A^*-algorithm) because it satisfies the triangular inequality [22,3]:

$$\text{for } a,b,c \in \mathbb{R} \qquad a + b \geq c \Rightarrow \lceil a \rceil + \lceil b \rceil \geq \lceil c \rceil$$

For a computational efficiency of the algorithm, we implement the v^*-visibility. Hence, at each pixel p in the buckets d, we associate a data structure that contains: its coordinates, the current source pixel p_i such that $v(p_i, p)$ and the distance $d_{geodes}(s, p_i)$.

We also have an obstacle data structure associated to each new source. Each obstacle list contains the set of obstacles sorted by polar angles met during the visibility propagation associated to each source.

We can know present the discrete geodesic algorithm. Note that some steps of this pseudo-code are not detailed for sake of clarity.

```
Discrete Geodesic Algorithm
Input: a domain D, a source s and a goal g
Output: the geodesic distance for each pixel of D

Let Bucket[i] be an array of FIFO queues
Let O[i] be an array of double-linked list of obstacles
Let d denotes the current bucket (d:=0)
Append_last(s,Bucket[d])
While there is no more pixel in buckets
    If the bucket d is empty then increment(d)
    t:=remove_first(Bucket[d])
    For each 8-neighbor n of t not labelled closed or visible
        If n is an obstacle then
            Add n to the obstacle list associated to the source of p
        else
            Let (u,l) be the localization of n in the sorted set O[i] associated to the current source
            If n is visible then
                Label n as visible
                Compute the geodesic distance d' of n
                Append_last(n,Bucket[d']) if d' > d
            else
                Label n as closed
                Initialization of new source n whose obstacle list is empty
                Compute the geodesic distance d' of n
                Append_last(n,Bucket[d']) if d' > d
    endFor
endWhile
```

4. Experiments and discussions

In our experiments, we compute the geodesic distance labelling of a binary image according to the coordinates of a source. In figure 3, we present the distance labelling with three metrics: d_4, d_8 and d_{geodes} in various domains. Geodesic distances are represented using a circular gray scale map in order to check the wave front propagation. In figures 4, instead of labelling the pixels according to their distance, pixels with the same color belong to the same equivalence class for the

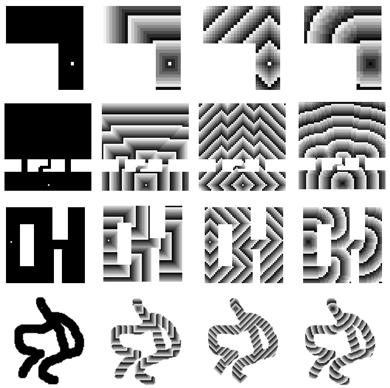

Figure 3. From the left column to the right column: the discrete domains and the source point (isolated white pixels), the geodesic labelling using d_8, the geodesic labelling using d_4, the geodesic labelling using d_{geodes}.

visibility problem. An illustration of these figures can be the minimum number of guards needed to control a room and the visibility associated to each guard (the first guard is given here). In figure 5, we present discrete geodesic metric on a blood vessel network. The domain is computed using a segmented angiography image.

Using this geodesic distance algorithm, we naturally would like to apply this algorithm to compute the discrete Voronoi diagram or the Euclidean distance transform just considering multiple sources. Since this algorithm use a local propagation scheme (as the Cuisenaire's algorithm [6]), the classical Danielsson's algorithm errors are not solved in this approach. Hence, this algorithm presents a solution to this problem but errors may occur.

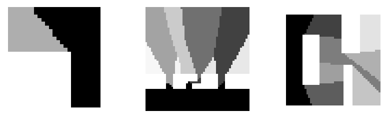

Figure 4. Global visibility graph: each pixel with the same color are in the same visibility equivalence class, source points of domains are the same of figure 3.

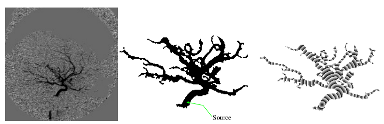

Figure 5. Application of the geodesic labelling in medical imaging: *left* An angiography image, *middle* binary image when blood vessels are segmented and *right* the geodesic labelling.

5. On an extension to 3D domains

A natural extension of these algorithms is to define both visibility and geodesic paths to higher dimensions and in particular to three dimensional domains. Hence, in the following we consider the domain $\mathcal{D}$ as a 26-connected set of voxels and $\bar{\mathcal{D}}$ the set of obstacle voxels.

In the case of mask based metrics, efficient algorithms exist for the geodesic distance since the problem can be shifted to a classical shortest path on a weighted graph as in 2D [16,10].

In our proposal of visibility based geodesic paths, definitions of the visibility are the same as in dimension 2, *i.e.*, two points in a 3D domain with obstacles are said to be visible if and only

if there exists at least one 3D discrete straight segment joining these points whose voxels belong to the domain. In the literature, many algorithms exists for drawing 3D lines between points: [4,15,1]. Based on an arithmetical definition of 3D discrete straight lines [12], recognition and segmentation of a 3D curve into digital straight segments algorithms exist [5].

For the visibility labelling in non-convex domains, the problem becomes more complex as in 2D. As the matter of fact, the parameter space analysis of 3D straight line is more complicated and the visibility test considering complementary of domains associated to voxels is not so direct. To illustrate problems, we consider the weak visibility in 3D. Hence, the visibility domain associated to two points s and t, is the set of straight lines going through s whose digitization contains t. Without loss of generality, we consider $[st] = (a, b, c)^T$ in the first $48th$ of space (*i.e.* $a \geq b \geq c$). So, in the primal space, the Euclidean 3D straight lines go through s and cross the square centered on t of size 1 (cf figure 5-a). If we consider now obstacles pixels, we consider the projection of squares on the yz-plane containing t. Then, the visibility domain is the set of lines going through s and crossing the projected square associated to t at a point not covered by obstacle projected square (cf figure 5-b). During this weak visibility test, the number of *blocking voxels* as in 2D cannot be bounded by a constant. Furthermore, the algorithm to detect if there exists a subset of the projected square of t which is not covered by blocking voxels is not evident.

Despite these difficulties, we can present a first solution based on a 3D generalization of the Soille's approach [25–27]. In that case, the visibility test between s and t is the following: we construct a 3D discrete straight segment between s and t using a 3D generalisation of Bresenham's algorithm [1,5] and we verify that all pixels of this segment belong to $\mathcal{D}$. Using this visibility labelling, we can use the same data structure as in the 2D case based on Bucket list and thus we can design a geodesic distance labelling algorithm. Note that the computational cost of this algorithm is high: the visibility test is done in $O(d)$ where d denotes the diameter of $\mathcal{D}$ and then

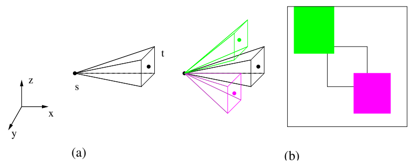

Figure 6. Illustration of the weak visibility ib 3D: (a) primal space illustration of the visibility domain, (b) visibility test, gray pyramids represent obstacle visibility domains and their projection on the yz-plane containing t.

we have a global cost in $O(nd)$ if n is the number of voxels in $\mathcal{D}$.

Figure 7 presents an illustration of the visibility labelling algorithm considering non-convex 3D domains. Figure 8 illustrates the geodesic distance labelling using a circular gray scale map. If we consider a discrete surface as a 3D domain where both inside and outside are obstacle voxels, we can compute geodesic distance on these objects (see figure 9).

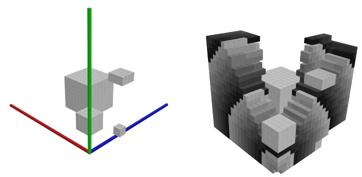

Figure 7. Visibility labelling :*left* obstacles of a 3D domain, *right* result of the visibility labelling where the source point is the lower corner.

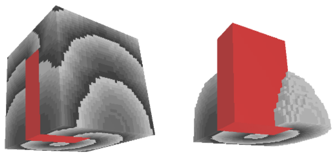

Figure 8. Geodesic labelling on a 3D domain with obstacle :*left* global labelling, *right* result when the distances are thresholded.

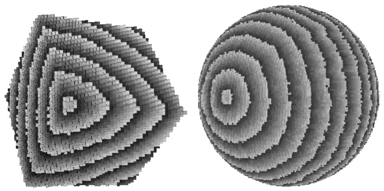

Figure 9. Geodsic distance labelling on surfaces: *left* rotated cube, *right* sphere.

6. Conclusion

In this article, we have presented a discrete definition of the visibility in classical computational geometry. This definition is based on well known discrete objects (DSS) and is computed only with integers. Based on this definition, we have presented several algorithms to solve several problems: if we want to decide if there exist DSS between two pixels, we have a cost linear in the number of obstacle pixels $O(m)$; if we want to label all pixels in a domain visible from a source, we have an algorithm in $O(nm)$. Using the weak visibility definition, we reduce the complexity of both algorithms respectively to $O(log(m))$ and $O(nlog(m))$. We also have presented a definition of discrete geodesic paths and an algorithm that compute the geodesic distance of each point in the domain according to a source.

This article also introduces open problems: is it possible to find an efficient data structure for the straightforward visibility algorithm ? How to generalize this approach for 3D domains and for discrete surfaces ? For this last problem, we have presented a first solution but more efficient algorithms similar to the 2D ones are expected.

REFERENCES

1. J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *Eurographic's 87*, pages 3–12, 1987.
2. G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, June 1986.
3. J.P. Braquelaire and P. Moreau. Error free construction of generalized euclidean distance maps and generalized discrete voronoï diagrams. Technical Report 84094, Université Bordeaux, Laboratoire LaBRI, 1994.
4. J.E. Bresenham. Algorithm for computer control of a digital plotter. In *IBM System Journal*, volume 4, pages 25–30, 1965.
5. D. Coeurjolly, I. Debled-Rennesson, and O. Teytaud. Segmentation and length estimation of 3d discrete curves. In *Digital and Image Geometry*. to appear, Springer Lecture Notes in Computer Science, 2001.
6. O. Cuisenaire. *Distrance Transformations : Fast Algorithms and Applications to Medical Image Processing*. PhD thesis, Universit Catholique de Louvain, oct 1999.
7. P.E. Danielsson. Euclidean distance mapping. *CGIP*, 14:227–248, 1980.
8. I. Debled-Rennesson. *Etude et reconnaissance des droites et plans discrets*. PhD thesis, Thèse. Université Louis Pasteur, Strasbourg, 1995.
9. I. Debled-Rennesson and J.P. Reveillès. A linear algorithm for segmentation of digital curves. In *International Journal of Pattern Recognition and Artificial Intelligence*, volume 9, pages 635–662, 1995.

10. G. Sanniti di Baja and S. Svensson. Detecting centres of maximal discs. *Discrete Geometry for Computer Imagery*, pages 443–452, 2000.
11. L. Dorst and A.W.M. Smeulders. Decomposition of discrete curves into piecewise straight segments in linear time. In *Contemporary Mathematics*, volume 119, 1991.
12. O. Figueiredo and J.P. Reveillès. A contribution to 3d digital lines. In *Proc. DCGI'5*, pages 187–198, 1995.
13. J. Françon, J.M. Schramm, and M. Tajine. Recognizing arithmetic straight lines and planes. *Discrete Geometry for Computer Imagery*, 1996.
14. A. Jonas and N. Kiryati. Digital representation schemes for 3d curves. *Pattern Recognition*, 30(11):1803–1816, 1997.
15. A. Kaufman and E. Shimony. 3-d scan conversion algorithms for voxel-based graphics. In *ACM Workshop on Interactive 3D Graphics, ACM Press, NY*, pages 45–75, 1986.
16. N. Kiryati and G. Székely. Estimating shortest paths and minimal distances on digitized three-dimension surfaces. *Pattern Recognition*, 26(11):1623–1637, 1993.
17. R. Klette and J. Zunic. Convergence of calculated features in image analysis. Technical Report CITR-TR-52, University of Auckland, 1999.
18. V. Kovalevsky and S. Fuchs. Theoritical and experimental analysis of the accuracy of perimeter estimates. In *Robust Computer Vision*, pages 218–242, 1992.
19. M. Lindenbaum and A. Bruckstein. On recursive, o(n) partitioning of a digitized curve into digital straigth segments. *IEEE Transactions on PatternAnalysis and Machine Intelligence*, PAMI-15(9):949–953, september 1993.
20. M. D. McIlroy. A note on discrete representation of lines. *Atandt Tech. J.*, 64(2, Pt. 2):481–490, February 1985.
21. N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, 31(1):114–127, January 1984.
22. P. Moreau. *Modélisation et génération de dégradés dans le plan discret*. PhD thesis, Université Bordeaux I, 1995.
23. J. Piper and E. Granum. Computing distance transformations in convex and non-convex domains. *Pattern Recognition*, 20:599–615, 1987.
24. I. Ragnemalm. *Contour processing distance transforms*, pages 204–211. World Scientific, 1990.
25. P. Soille. Spatial distributions from contour lines: an efficient methodology based on distance transformations. *Journal of Visual Communication and Image Representation*, 2(2):138–150, June 1991.
26. P. Soille. Generalized geodesy via geodesic time. *Pattern Recognition Letters*, 15(12):1235–1240, December 1994.
27. P. Soille. *Morphological Image Analysis*. Springer-Verlag, Berlin, Heidelberg, New York, 1999.
28. B. J. H. Verwer, P. W. Verbeek, and S.T Dekker. An efficient uniform cost algorithm applied to distance transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(4):425–429, April 1989.
29. B.J.H Verwer. Local distances for distance transformations in two and three dimensions. *Pattern Recognition Letters*, 12:671–682, november 1991.
30. J. Vittone and J.M. Chassery. Recognition of digital naive planes and polyhedization. In *Discrete Geometry for Computer Imagery*, number 1953 in Lecture Notes in Computer Science, pages 296–307. Springer, 2000.
31. L.D. Wu. On the chain code of a line. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 4:347–353, 1982.