Conceptualization of seeded region growing by pixels aggregation. Part 2: how to localize a final partition invariant about the seeded region initialisation order.

Vincent Tariel

***Abstract*— In the previous paper, we have conceptualized the localization and the organization of seeded region growing by pixels aggregation (SRGPA) but we do not give the issue when there is a collision between two distinct regions during the growing process. In this paper, we propose two implementations to manage two classical growing processes: one without a boundary region region to divide the other regions and another with. Unfortunately, as noticed by Mehnert and Jakway (1997), this partition depends on the seeded region initialisation order (SRIO). We propose a growing process, invariant about SRIO such as the boundary region is the set of ambiguous pixels.**

***Index Terms*— Boundary, mathematical morphology, Minkowski addition, seeded region growing by pixel aggregation.**

I. Introduction

In the previous paper[5], we have conceptualized the localization and the organization of seeded region growing by pixels aggregation (SRGPA). This conceptualization has permitted to create a library dedicated to the implementation of algorithm using SRGPA (see annexe I). Each implementation using this library is quick and provides efficient algorithms.
At the end of most of algorithms using SRGPA, the regions are a partition of the space. In a classical growing process, there are two possible approaches to partition the space: one without a boundary region to divide the other regions, another with. Thanks to the conceptualization of SRGPA, it is easy to implement an algorithm such as the regions respect one or the other partitions at the end of the growing process. However, this partition depends on the seeded region initialisation order (SRIO)[1], [2]. The localization of the inner border of each region depends on the SRIO. To overcome this problem, we define a set of ambiguous points. This set is called ambiguous points because in discrete space, there are some points such as it is impossible to determine to which regions they belong. We define a growing process that affect:

- the no ambiguous points to the appropriate regions,
- the ambiguous points to the boundary region.

In this article, the notations are:

- let E be a discrete space[1],
- let Ω be a domain of E and I its characteristic function such as $\Omega = \{\forall x \in E : I(x) \neq 0\}$,

[1] The space E, is a n-dimensional discrete space $\mathbb{Z}^n$, consisting of lattice points which coordinates are all integers in a three-dimensional Euclidean space $\mathbb{R}^n$. The elements of a n-dimensional image array are called points.

Using this growing process, the localization of final partition is invariant about the SRIO.
The outline of the rest of the paper is as follows: in Sec. II, we present the two classical growing processes. In Sec. III, we explain how to implement a growing process invariant about the SRIO. In Sec. V, we make concluding remarks.

II. Classical growing processes

This section presents two classical growing processes. For the first, there is no boundary region to divide the other regions. For the second, there is a boundary region to divide the other regions. The geodesic dilatation[4] is used like an example but this approach can be used for the most of algorithms using SRGPA if the algorithm can be reduced in a succession of geodesic dilatations[3]. This section is decomposed in two parts: definition of two distinct partitions and how to get both partitions for algorithms using SRGPA.

A. Two distinct partitions

A segmentation of Ω is a simple-partition of Ω into subsets X_i , $i = 1, \dots, m$, for some m if:

1) $\Omega = \cup_{i=1}^{m} X_i$
2) $\forall i \neq j \Rightarrow X_i \cap X_j = \emptyset$

A segmentation of Ω is a V-boundary-partition[2] of Ω into subsets X_i $i = 1, \dots, m$, for some m, and X_b if:

1) $\Omega = (\cup_{i=1}^{m} X_i) \cup X_b$
2) $\forall i \neq j \Rightarrow (X_i \oplus V) \cap X_j = \emptyset$
3) $X_b \ominus V = \emptyset$

The second condition defines that the boundary region divides the other regions and the third condition defines that the boundary region thickness is equal to 1.

B. Simple-partition

To get a simple-partition using the SRGPA, the zone of influence (ZI) at each region is localized on the outer boundary region excluding all other regions: $Z_i^t = (X_i^t \oplus V) \setminus (\bigcup_{j \in \mathbb{N}} X_j)$.
During the growing process, when a couple (x, i) is extracted from the SQ, there is a simple growth: p.growth(x, i)). At the end of the growing process, the regions $X_i^{t=\infty}$ $i = 1, \dots, m$ are a simple-partition of Ω. The algorithm 1 is an example (see figure 1).

[2] A V-boundary-partition is also a simple-partition.

Algorithm 1 Geodesic dilatation

Require: I, S , V *//The binary image, the seeds, the neighbourhood*
// initialization
System_Queue s_q($\delta(x,i) = 0$ if $I(x) \neq 0, OUT$ else, FIFO, 1); *//A single FIFO queue such as if $I(x) = 0$ then (x,i) is not pushed in the SQ.*
Population p (s_q); *//create the object Population*
Restricted $N = \mathbb{N}$;
Tribe active(V, N);
. $\forall s_i \in S$ **do**
  int ref_tr = p.growth_tribe(actif); *//create a region/ZI, (X_i^t, Z_i^t) such as* $\boxed{Z_i^t = (X_i^t \oplus V) \setminus (\bigcup_{j\in\mathbb{N}} X_j)}$
  p.growth(s_i, ref_tr);
end for
//the growing process
s_q.select_queue(0); *//Select the single FIFO queue.*
while s_q.empty()==false **do**
  $(x,i) = s\_q$.pop();
  p.growth(x, i);
end while
return p.X();

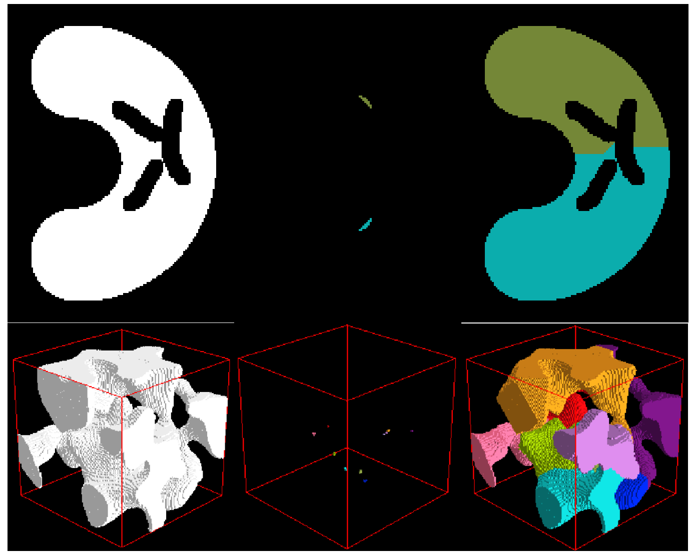

Fig. 1. For both series, the first image is the initial image, the second image is the seeds and the last image is the simple-partition after the geodesic dilatation. The first serie is the case in 2D and the second in 3D. For both, the regions are a simple-partition of $\Omega = \{\forall x \in E : I(x) \neq 0\}$

C. The V-boundary-partition

To get a simple-partition using the SRGPA, a boundary region, X_b, is added such as its ZI is always empty. For all the regions except the boundary region, their ZI are localized on the outer boundary region excluding all the regions: $Z_i^t = (X_i^t \oplus V) \setminus (\bigcup_{j\in\mathbb{N}} X_j)$. The simple growth, p.growth(x, i), is substituted by

- if there is more than 2 ZI on x, then growth on x of the boundary region,
- else the growth on x of the region i

Using this definition, at the end of the growing process, the regions $X_i^{t=\infty}$ $i = 1, \ldots, m$, and X_b are a V-boundary-partition of Ω. The algorithm 2 is an example (see figure 2).

D. The partition depends on SRIO

Whatever the growing process is, the final partition is not invariant about SRIO. The figure 3 shows the case with an ambiguous pixel for the growing process without a boundary region to divide the other regions. The figure 4 shows the case with two ambiguous pixels for the growing process with a boundary region to divide the other regions. The localization of the inner border of each region depends on SRIO. The next section proposes a solution to overcome this limitation.

III. Invariance about the seeded initialisation order

A. Why is there dependence?

Definition Let Ω be a domain of E and a and b two points of Ω. We call geodesic distance $d_\Omega(a,b)$ in A the lower bound

Algorithm 2 Geodesic dilatation with an boundary

Require: I, S , V *//The binary image, the seeds, the neighbourhood*
// initialization
System_Queue s_q($\delta(x,i) = 0$ if $I(x) \neq 0, OUT$ else, FIFO, 1); *//A single FIFO queue such as if $I(x) = 0$ then (x,i) is not pushed in the SQ.*
Population p (s_q); *//create the object Population*
Tribe passive($V = \emptyset$);
//create a boundary region/ZI, (X_b^t, Z_b^t) such as $Z_i^t = \emptyset$
int ref_boundary = p.growth_tribe(passive);
Restricted N=$\mathbb{N}$;
Tribe active(V, N);
. $\forall s_i \in S$ **do**
  int ref_tr = p.growth_tribe(actif); *//create a region/ZI, (X_i^t, Z_i^t) such as* $\boxed{Z_i^t = (X_i^t \oplus V) \setminus (\bigcup_{j\in\mathbb{N}} X_j)}$
  p.growth(s_i, ref_tr);
end for
//the growing process
s_q.select_queue(0); *//Select the single FIFO queue.*
while s_q.empty()==false **do**
  $(x,i) = s\_q$.pop();
  if pop.Z()[x].size()≥2 **then**
    //growth of the boundary region
    p.growth(x, ref_boundary);
  else
    p.growth(x, i); *//simple growth*
  end if
end while
return p.X();

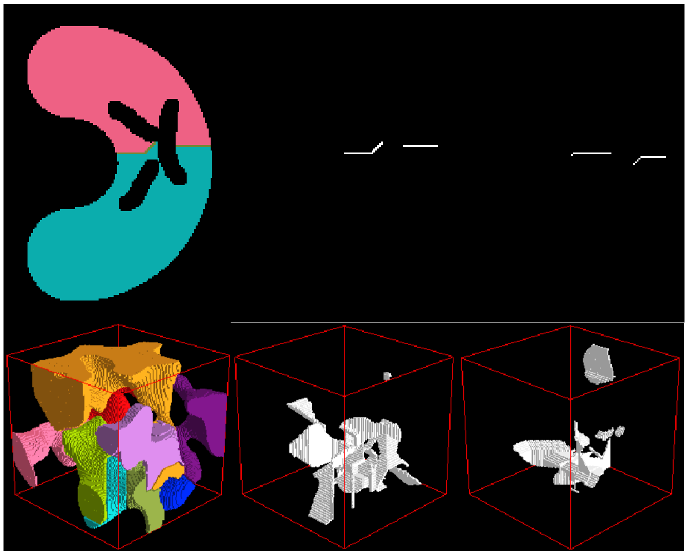

Fig. 2. For both series, the first image is V-boundary-partition obtained by the geodesic dilatation with an boundary, the second figure and third figure are the visualisation of the boundary region depending on the choosen neighbourhood. For the second figure, it is the 8-neightborhood in 2D and 26-neightborhood in 3D and for the third figure, it is 4-neightborhood in 2D and 6-neightborhood in 3D.

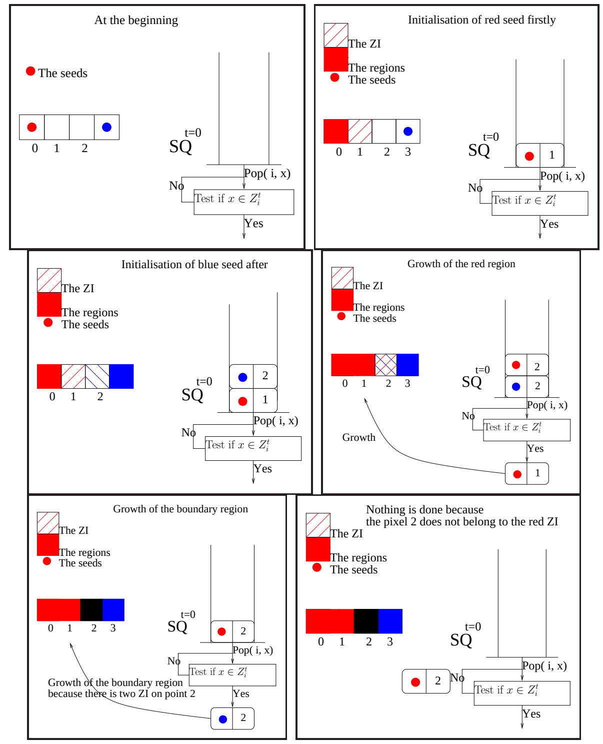

Fig. 4. This serie shows the geodesic dilatation with a boundary region to divide the other regions such as the red seed is initialized firstly. The point 1 and 2 are ambiguous pixels in this growing process because they belong to different region depending on SRIO. In this case, the point 1 belongs to the red region and the point 2 belongs to the boundary region but if the blue region is initialized at first, the point 1 will belong to the boundary region and the point 2 will belong to the blue region.

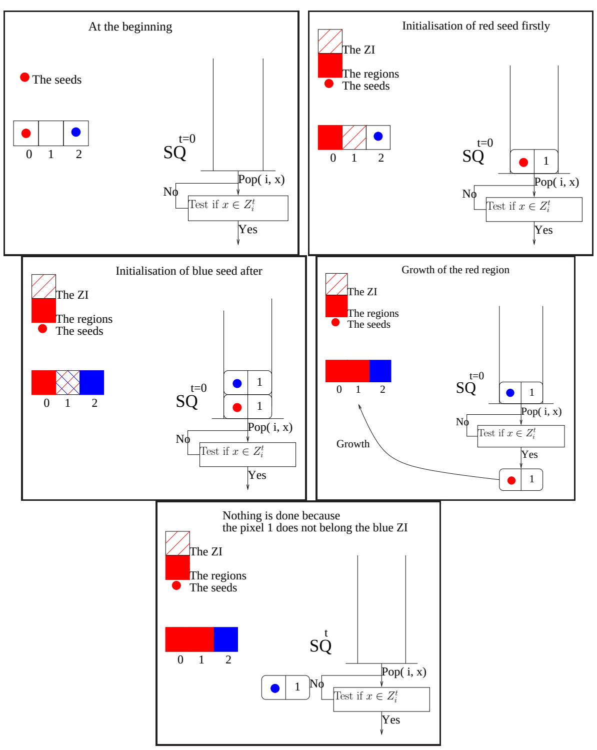

Fig. 3. This serie shows the geodesic dilatation without a boundary region to divide the other regions such as the red seed is initialized firstly. The point 1 is an ambiguous point in this growing process because this point belongs to the region initialized firstly. In this case, it is the red region.

of the length of the paths y in Ω linking a and b.

Let s be a set. We call the geodesic distance $d_\Omega(s,b) = \min_{\forall a \in s} d_\Omega(a,b)$, the lower bound of all geodesic distance $d_\Omega(a,b)$ such as a belongs to s.

The geodesic influence zone[4], $z_A(s_i)$, of the seeds, $S = (s_i)_{1\leq i\leq n}$, of E in Ω, is the set of the points of Ω, for which the geodesic distance to s_i is smaller than the geodesic distance to other seeds of S.

$$z_A(s_i) = \{\forall x \in \Omega : (\forall j \neq i \Rightarrow d_\Omega(s_i) < d_\Omega(s_j)\}$$

The $z_A(s_i)_{1\leq i\leq n}$ is not a partition of Ω because $\cup_{i=1}^{n} z_A(s_i) \neq \Omega$. In fact, it is possible to demonstrate that $\cup_{i=1}^{n} z_\Omega(s_i) = \Omega \uplus A$. The symbol $\uplus$ means the disjoint union:$B \uplus C = \{B \cup C : B \cap C = \emptyset\}$. The set A, called ambigous points, is

$$\begin{gathered} A = \{\forall x \in \Omega : \exists i,j \Rightarrow \\ (d_\Omega(s_i) = d_\Omega(s_j)) \text{ and} \\ ((\forall k \neq \text{(i and j)}) : d_\Omega(s_i) \leq d_\Omega(s_k))\} \end{gathered}$$

The set$A$ is all the points of Ω for which the geodesic distance to s_i and $s_{j\neq i}$ is equal and smaller than the geodesic distance to other seeds of S. The $z_A(s_i)_{1\leq i\leq n}$ and A is a simple-partition of Ω. In the previous implementation of the geodesic dilatation, the ambiguous points are distributed depending on the seeded initialisation order (see figure 3 and 4). The next paragraph presents an implementation such as the boundary region is the set of ambiguous points.

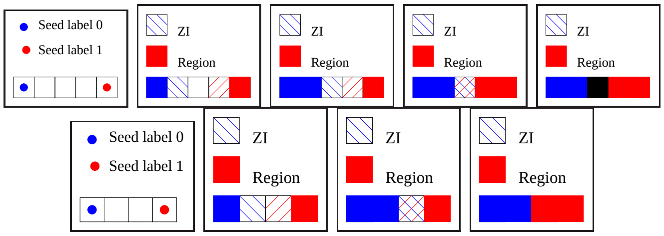

Fig. 5. The first serie is the case of one ambiguous point. There is a classical growth until there are two ZI in the same point, x. In this case, the min_elements(pop.Z()[x]) returns 0 because there are two ZI of label 0 and 1. There is the boundary growth because the couple extracted from the queue has a label 0 equal to min_elements(pop.Z()[x]). The second serie is the case without ambiguous point. There is a classical growth until there is two ZI in the same point, x. The min_elements(pop.Z()[x]) returns 0 because there are two ZI of label 0 and 1. There is the region growth of label 1 because the couple extracted from the queue has a label 1 not equal to min_elements(pop.Z()[x]).

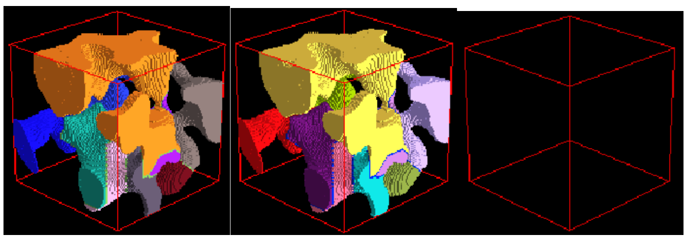

Fig. 6. The two first images are the geodesic dilatation with a boundary region localized on the ambiguous points such as the SRIO is different. The third image represents the boundary difference of the two previous images. It is empty image since the region localization at the end the growing process is invariant about the SRIO.

B. Boundary as ambiguous points

We suppose in this paragraph that the seeded initialisation follows this order $0, 1, \ldots, n$.
To get a boundary localized on the ambiguous points using the SRGMPA, a boundary region is added such as its ZI is always empty. For all the regions except the boundary region, their ZI are localized on the outer boundary region excluding all the regions: $Z_i^t = (X_i^t \oplus V) \setminus (\bigcup_{j\in\mathbb{N}} X_j)$. When a couple (x, i) is extracted from the SQ, there is (see figure [5,6] and algorithm 3):

1) p.growth(x, boundary region)) if there is more than two ZI in x and if i = min_elements(pop.Z()[x]),
2) p.growth(x, i)) else

This partition is invariant about the SRIO but is not a V-boundary-partition (see figure 5) since there are some holes on the boundary region.

IV. Conclusion

In discrete space, the boundary definition is not oclearly defined. Using the SRGPA, we have proposed two growing processes to do a simple or V-boundary partition. These growing processes have incertitude on the regions boundary localisation. To overcome this problem, we have defined a set of ambiguous points such as in a discrete space, it is impossible to know to which regions they belong. Knowing that, we have defined a growing process with a boundary region localized

Algorithm 3 Geodesic dilatation with a boundary as ambiguous points

Require: I, S , V *//The binary image, the seeds, the neighborhood*
// initialization
System_Queue s_q($\delta(x, i) = 0$ if $I(x) \neq 0, OUT$ else, FIFO, 1); *//A single FIFO queue such as if $I(x) = 0$ then (x, i) is not pushed in the SQ.*
Population p (s_q); *//create the object Population*
Tribe passive($V = \emptyset$);
//create a boundary region/ZI, (X_b^t, Z_b^t) such as $Z_i^t = \emptyset$
int ref_boundary = p.growth_tribe(passive);
Restricted N=ℕ;
Tribe active(V, N);
. $\forall s_i \in S$ in the order $0, 1 \ldots$ **do**
  int ref_tr = p.growth_tribe(actif); *//create a region/ZI, (X_i^t, Z_i^t) such as* $\boxed{Z_i^t = (X_i^t \oplus V) \setminus (\bigcup_{j\in\mathbb{N}} X_j)}$
  p.growth(s_i, ref_tr);
end for
//the growing process
s_q.select_queue(0); *//Select the single FIFO queue.*
while s_q.empty()==false **do**
  $(x, i) = s\_q$.pop();
  if pop.Z()[x].size()≥2 and i= min_elements(pop.Z()[x]) **then**
    p.growth(x, ref_boundary); *//growth of the boundary region*
  else
    p.growth(x, i); *//simple growth*
  end if
end while
return p.X();

on these ambiguous points. The associated partition to this growing process is invariant about the SRIO but it is only a simple since there are some holes on the boundary region. Depending on the algorithm or the application, it is possible to apply a post-treatment to label these ambiguous points to the regions. For example in the case of the evolution of the cement paste microstructure, the ambiguous pixels have been always affected to the void phase. There is an over-localization of this phase but the error due to the over-localization is always the same and can be estimated.

Appendix I
Summary of the previous article

The idea of the first article is to define three objects: Zone of Influence (ZI), System of Queues (SQ) and Population. The algorithm implementation using SRGPA is focused on the utilisation of these three objects. An object ZI is associated to each region and localizes a zone on the outer boundary of its region. For example, a ZI can be the outer boundary region excluding all other regions. An algorithm using SRGPA is not global (no treatment for a block of pixels) but local (the iteration is applied pixel by pixel belonging to the ZI). To manage the pixel by pixel organisation, a SQ sorts out all pixels belonging to ZI depending on the metric and the entering time. It gives the possibility to select a pixel following a value of the metric and a condition of the entering time. The object population links all regions/ZI and permits the (de)growth of regions. A pseudo-library, named Population, implements these three objects. An algorithm can be implemented easier and faster with this library, fitted for SRGPA.

Acknowledgment

I would like to thank my Ph.d supervisor, P. Levitz, for his support and his trust. The author is indebted to P. Calka for valuable discussion and C. Wiejak for critical reading of the manuscript. I express my gratitude to the Association Technique de l'Industrie des Liants Hydrauliques (ATILH) and the French ANR project "mipomodim" No. ANR-05-BLAN-0017 for their financial support.

References

[1] S. Beucher. Algorithmes sans biais de ligne de partage des eaux. *Note interne CMM*, 2004.

[2] A. Mehnert and P. Jackway. An improved seeded region growing algorithm. *Pattern Recognition Letters*, 18(10):1065–1071, October 1997.

[3] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *Ieee Transactions On Pattern Analysis And Machine Intelligence*, 18(12):1163–1173, December 1996.

[4] M. Schmitt. Geodesic arcs in non-euclidean metrics: Application to the propagation function. *Revue &Intelligence Artificielle*, 3, no.2:43–76, 1989.

[5] V. Tariel. Conceptualization of seeded region growing by pixels aggregation. part 1: the framework. *submitted*, 2008.