Image Anal Stereol ?? (Please use \volume):?? (Please use \startpage)-7
Original Research Paper

ULTIMATE OPENING IMPLEMENTATION BASED ON A FLOODING PROCESS

Retornaz Thomas and Marcotegui Beatriz

Centre de Morphologie Mathématique. Ecole des Mines. Fontainebleau, France
e-mail: retornaz,marcotegui@cmm.ensmp.fr
(Accepted ?? (Please use \accepted*))*

ABSTRACT

The ultimate opening is an operator based on numerical residues. It extracts the significant structures from an image, without a priori size information. Its direct implementation, derived from its definition is time-consuming. In this paper we propose an efficient implementation of ultimate criteria opening, based on a flooding process.

Keywords: numerical residue, ultimate criteria opening, mathematical morphology, implementation.

ULTIMATE OPENING

Ultimate opening, introduced by (Beucher, 2005; 2007), is a residual operator where primitives are respectively an opening of size i and an opening of size i+1. In other words, it analyzes the evolution of an image with a family of openings of increasing sizes. The difference (residues) between two successive openings is considered. The maximum of these residues (for each pixel) is kept and two pieces of information are stored:

- R_θ, the maximum residue, that conveys contrast information
- q_θ, the size of the opening producing the maximum residue, that contains the size information of the corresponding structure.

The ultimate opening is then defined as follows:

$$\begin{aligned} &R_\theta(I) = sup(r_\lambda(I)), \forall \lambda \geq 1 : r_\lambda(I) = \gamma_\lambda - \gamma_{\lambda+1} \\ &q_\theta(x) = max(\lambda) : \lambda \geq 1, r_\lambda(x) = R_\theta(x) \text{ and } > 0 \end{aligned} \quad (1)$$

In this paper we focus on attribute (criteria) openings (see Breen and Jones (1996)). These are openings by reconstruction, associated to a given criterion, the simplest ones being area, width, height or diameter = max(width,height)[1]. These criteria are illustrated in figure 1 for a given set A.

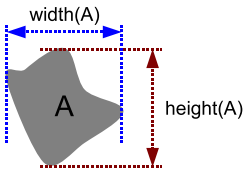


Figure 1. *Example of criteria : width, height or diameter = max(width,height).*

See figure 2 for an example of ultimate closing (height criterion). Contrasted structures appear with high value in R_θ and q_θ is a pre-segmentation of the image, each pixel valuated with the size of the structure containing it.

Successful applications have been built on the ultimate opening: a text localization approach, ranked first in the ImagEval evaluation campain (`http://www.imageval.org`) and a rock granulometry introduced by Outal (2006).

DIRECT IMPLEMENTATION

The pseudocode of the direct implementation (derived from equation 1) is presented in algorithm 1 in page 5.

This implementation is generic (valid for any γ) but it is time consuming and makes the approach useless for some applications, in particular with large images.

An efficient implementation of attribute openings based on a flooding process has been proposed in Breen and Jones (1996). In the following section we introduce the modifications required in order to derive an efficient implementation of the ultimate criteria opening. This ultimate opening implementation is restricted to attribute openings.

[1]Width and height correspond to horizontal and vertical Feret diameter

IMPLEMENTATION BASED ON A FLOODING PROCESS

In order to obtain an ultimate closing (by duality an ultimate opening) a flooding of the image is simulated. We analyze the residue of a lake (the flood level increase) each time it floods a new level. The ultimate closing keeps the maximum of these residues for each pixel.

In order to illustrate the idea, let's follow, step by step, the ultimate area closing of fig 3. Flooding starts from the two minima, generating two lakes: $L_1 = \{b\}$ and $L_2 = \{f,g\}$. The first residue is generated on pixel b when L_1 floods pixel c (first step): $R_\theta\{b\}$ = Current Level − Previous level = $3-2=1$ and $q_\theta\{b\}=2$ (previous size $+1$)[2] because this residue appears for a closing of size 2.

Then L_2 floods pixel e (step 2) leading to the valuation of pixels f and g : $R_\theta\{f,g\}=4-1=3$ and $q_\theta\{f,g\}=2+1=3$.

Then, at level 5 (step 3), L_1 and L_2 are merged. A residue of 2 appears for L_1. This residue is larger than the previous one at pixel $\{b\}$, therefore its value should be updated: $R_\theta\{b\}=2$, $q\{b\}=3$. Note that q_θ is valuated with the criterion before merging + 1[3] and not the criterion value of the merged lake. Pixels c, e and h are also updated $R_\theta\{c\}=2$, $q\{c\}=3$, $R_\theta\{e,h\}=1$, $q_\theta\{e,h\}=5$. For $\{f,g\}$ current residues are smaller than $R_\theta\{f,g\}$, therefore their values are not modified. Lakes L_1 and L_2 have been merged and the flooding process continues.

At level six (Step 4), a residue of 1 appears. For pixels $\{b,c,f,g\}$ this value is smaller than the current maximal residues R_θ, therefore their values are not modified. For pixels $\{e,h\}$ this valuation is equal to R_θ. According to equation 1, q_θ should be updated: $q_\theta\{e,h\}=9$. Pixels $\{d\}$ and $\{i\}$ have not been processed yet: $R_\theta\{d,i\}=1$ and $q_\theta\{d,i\}=9$.

The flooding process stops at the maximum of the image, pixels $\{a,i\}$, are not valuated.

(a) Original Image

(b) Morphological Gradient on luminance

(c) R_θ

(d) q_θ Randomized

Figure 2. *Ultimate closing of a gradient image (height criterion). Image DataBase (LUCAS* et al.*, 2005).*

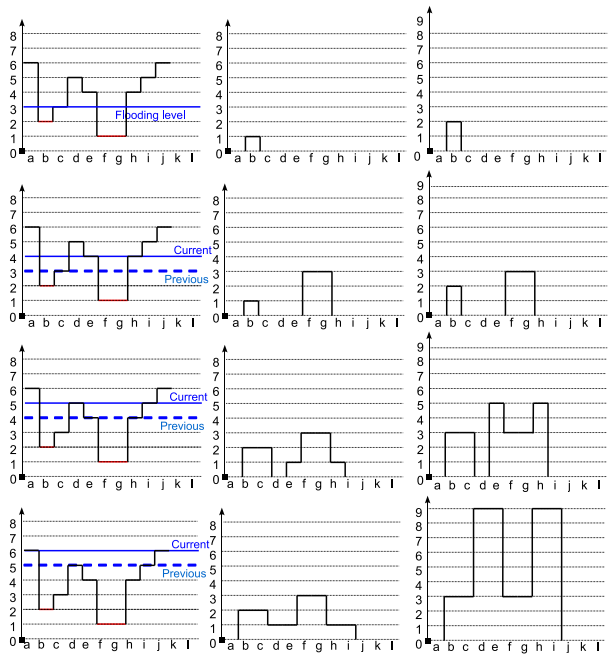


Figure 3. *From top to down steps of the flooding-based ultimate closing computation. From left to right: input image,* R_θ *and* q_θ*.*

The previous implementation seems quite straight forward. This is because the area criterion increases each time a new pixel is added to a lake. Particular situations appear when this is not the case.

Attribute closing of size λ guarantees that all minima have a criterion size larger than λ. If, for a given lake, a pixel of a plateau is required to verify this condition, the whole plateau is flooded by this lake (attribute closing is a connected operator).

Even if we perform the flooding process pixel by pixel, plateaus require a particular attention. Consider a lake of size PrevCriterion that floods a plateau. Two different situations may appear:

– the inclusion of the whole plateau does not modify the criterion (example plateau 6 of

[2]Note that, "+1" is correct for all criteria used in this paper. You may have to modify this for other criterion
[3]Idem

figure 4, described hereafter). The level should not be considered for residue computation.

– the inclusion of the plateau increases the criterion by K. The residue is updated with an associated q_θ = 1+PrevCriterion and no other residue will appear for this lake until q_θ = PreviousCriterion+K.

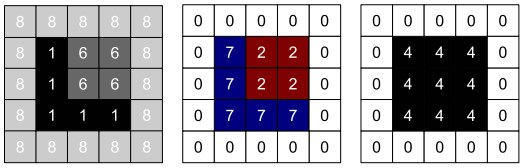
A residue is generated in a lake when both, flooding level and lake criterion, increase. Note that a flooding increase may not induce a criterion increase (see figure 4) and that the criterion may increase in the middle of a plateau (see figure 5). Even if these events happen separately, both are necessary to generate a residue in a lake.

Let's consider for example the width closing of figure 4. This image contains only 1 minimum (pixels with value = 1) of width=3. Closings of $\lambda \leq 3$ do not modify the image because all lakes already have a $width \geq \lambda$. A closing of $\lambda(width) = 4$ will result in an image of constant graylevel (8). Thus, flooding level 6 does not generate any residue, because the width of the lake remains equal to 3. In other words, a closing of width 3 does not modify the image while a closing of width 4 floods simultaneously levels 6 and 8. A residue of value 7 (resp. 2) should be ascribed to pixels of value 1 (resp. 6). Therefore residues should not be updated at each flooding level. If the residue had been updated at level 6, a residue of value 6 - 1 = 5 (instead of 7) would have been generated. Thus, residues should be updated only on lakes which criterion measure increases due to the flooding process. Otherwise, R_θ is underestimated and may lead to a wrong valuation of q_θ.

Rule 1: residues should be updated, only for lakes which criterion changes during the flooding process.

When flooding reaches a plateau, a residue is updated only if it induces a criterion increase. This may occur in the middle of the plateau. See dashed pixel 4 of figure 5 for an example: when flooding this pixel, the lake width increases from 2 to 3 and the residue is updated. When flooding the rest of the plateau, the lake criterion increases again. Even if the residue is not updated again (because the flooding level does not change within the plateau), the lake criterion (PrevCriterion) is systematically updated (line 39 in page 6). The aim of that is to know the correct lake criterion for further residue updating.

[4]The test has been released in P4© at 2.4 Ghz



(a) Image (b) R_θ (c) q_θ

Figure 4. *Update residue only if criterion increases*

Rule 2: Criterion should be systematically updated.

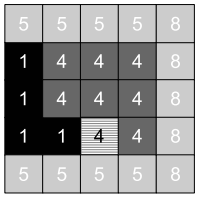


Figure 5. *Criterion should be systematically updated*

A maximal closing size may be considered (MaxCriterion), if the size of interesting structures is bounded. In that case, only residues that appear for a criterion smaller than MaxCriterion are considered. This parameter may be necessary if we want to extract structures nested in a more contrasted structure. Given that ultimate closing only takes into account the maximum residue, the process should be stopped before reaching the size of the containing structure.

The algorithm presented in section 6 p. 4 achieves all the rules described above.

PERFORMANCE OVERVIEW

The provided algorithm has been implemented in standard c++, without taking any special care about its speed. [4].

The performance gap between the basic implementation (Algorithm 1) and our proposed implementation (Algorithm 2) is necessarily huge. Because the first algorithm depends linearly on *MaxCriterion*. For example, the time to compute an ultimate closing on Figure 2(b) using $MaxCriterion = 100$ with the first algorithm is a hundred times longer than with the second one. So we will only discuss here the performances of our proposed algorithm.

To test the sensitivity of our algorithm to the image complexity in a controlled manner, we used

the protocol defined by Meijster and Wilkinson (2002). We have to generate synthetic images: N_{min} randomly placed dots on a black background were generated. After this, Euclidean distance maps were generated from theses. These distance images allow a controlled testing of the algorithm performance as a minima number function of roughly circular shape.

The evolution of CPU times for ultimate closings (height criterion) with the *MaxCriterion* is represented on Figure 6. The tests were processed on two 512x512 distance map images with different numbers of local minima M (Relies to N_{min}). The image corresponding to the dashed curved has a lot of minima – hence fewer gray levels – and we see that time tends to be constant. On the other hand the solid line has a quadratic shape and corresponds to an image with only two minima – hence many more gray levels.

We could observe here that the proposed algorithm seems to depend mostly on the number of gray levels L in the image. This dependency relies on Rule 1 defined page 2: the residues are updated for lakes which criterion changes during the flooding process.

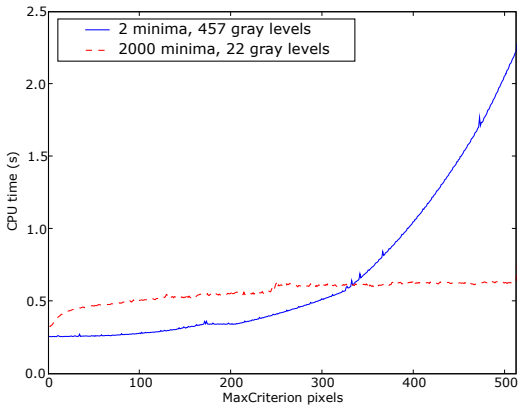


Figure 6. *CPU times on synthetic images: algorithm dependance on gray level numbers.*

To measure the dependence of CPU times on image size N, distance map images of different sizes with the same density α of minima (i.e $\alpha = N_{min}/N = constant$) were used. The results are shown in Figure 7, *MaxCriterion* is always fixed at the maximum allowed value (in this case height image size). In this case the algorithm seems to be linear in N.

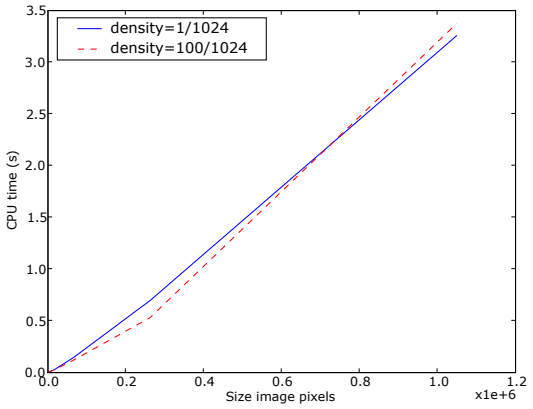


Figure 7. *CPU times on synthetic images: algorithm dependance on the image size.*

We provided also in Figure 8, the evolution of CPU times for ultimate closing (height criterion) with the *MaxCriterion* on natural images. More precisely, we provide mean and standard deviation cpu times measured on luma gradient over the 38 1600x1200 images of ICDAR database. This test was used to assess the performance under more realistic conditions.

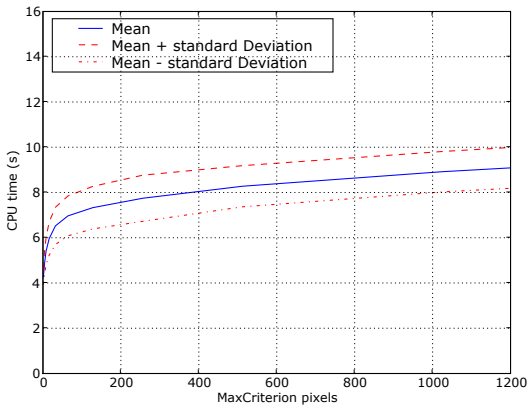


Figure 8. *CPU times on natural image: evolution of cpu time on natural images according to the growth of MaxCriterion parameter. Mean and standard deviation time over the 38 images (1600x1200) of ICDAR database measured on luma gradient using height criterion.*

CONCLUSION

In this paper we propose an efficient implementation of the ultimate criteria closing, based on a flooding process. This implementation requires criterion computation for each pixel

processing, which may be costly for complex criteria. In practice we use criteria that may be computed with a simple arithmetic operation. If the criterion requires complex computation the algorithm may be modified in order to compute the criterion for each flooding level instead of each pixel. In a rough manner, the proposed algorithm seems to depend mostly, on N (number of pixels of the image) and L (the number of gray levels) but we have also to integrate the merging cost. So a truly theoretical complexity analysis have to be done to provide an upper bound complexity of the proposed algorithm and an efficiency comparison with a max-tree (see Salembier *et al.* (1998)) and component-tree (see Najman and Couprie (2006); Berger *et al.* (2007)) based implementation is foreseen.

PSEUDOCODE ALGORITHM

ImRelief contains the input image and imLabels its labelled minima. The following variables are required during the flooding process in order to update the residues:

- the flooding level (prio-ws)
- the flooding level of each pixel the last time its residue has been updated. The input image imRelief is modified to store this information.
- for each lake, minimum imRelief value of all its pixels (PrevLevel).
- for each lake, previous lake criterion (PrevCriterion).
- for each lake, the list of pixels belonging to it (LakePixelList)

At the initialization step neighbor pixels of minima are introduced in the FAH and the criterion of each lake is initialized. Then a classical flooding process is simulated. The following modifications are required to derive an ultimate closing from the flooding process. An equivalence array is used in order to know if two lakes have already been merged or not. During the propagation step, a pixel (v) is extracted from the FAH and the LakeGrowing function is called. This function updates the residue of the lake if its criterion increases and if the flooding level is higher than the last time this residue has been updated. Note that PrevLevel is updated only if residue is updated (lines 36 to 38) but PrevCriterion is systematically updated (line 39).

Then neighbors of flooded pixels are considered:

Candidate neighbors are introduced in the FAH. If different label is found, v is a meeting point and LakeMerging function is called. This function updates the residue of both lakes (if required) and updates the level, the criterion and the list of pixels of the merged lake.

REFERENCES

Beucher S (2005). Numerical residues. In: Ronse C, Najman L, Decencière E, eds., Mathematical Morphology: 40 Years On, vol. 30 of *Computational Imaging and Vision*. Dordrecht: Springer-Verlag, 23–32.

Beucher S (2007). Numerical residues. Image and Vision Computing 25:405–15.

Breen EJ, Jones R (1996). Attribute openings, thinnings and granulometries. P.Maragos R.Schafer M.Butt Mathematical Morphology and its applications to image and signal processing 64:377–89, 1996.

Lucas S, Panaretos A, Sosa L, Tang A, Wong S, Young R, Ashida K, Nagai H, Okamoto M, Yamamoto H, Miyao H, Zhu J, Ou W, Wolf C, Jolion J, Todoran L, Worring M, and Lin X . Icdar 2003 robust reading competitions:entries, results, and future directions. International Journal on Document Analysis and Recognition, 7(2-3):105–122, 2005.

Meijster A, Wilkinson M.H.F. A comparison of algorithms for connected set openings and closings. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(4):484–494, 2002.

Najman L, Couprie M. Building the component tree in quasi-linear time. IEEE Transactions on Image Processing, 15(11):3531–3539, 2006.

Outal S (2006). Quantification par analyse d'images de la granulométrie des roches fragmentées : amélioration de l'extraction morphologique des surfaces,amélioration de la reconstruction stéréologique. Thèse de doctorat en morphologie mathématique et géosciences, ENSMP.

Salembier P, Oliveras A, Garrido L. Anti-extensive connected operators for image and sequence processing. IEEE Transactions on Image Processing, 7(4):555–570, 1998.

Berger C, Géraud T, Levillain R,Widynski N. Effective component tree computation with application to pattern recognition in astronomical imaging. *In the proceedings of the IEEE International Conference on Image Processing, San Antonio, Texas, USA*, pages 16–19, September 2007.

Data: *imRelief* Input image, *step* (step of increasing openigns; default=1),*MaxCriterion* (maximum opening size)
Result: R_θ (*Transformed* image), q_θ (*Associated* image)

1 • ***Initialisation*** **:**
2 $R_\theta \longleftarrow 0$; $q_\theta \longleftarrow 0$
3 imPreviousOpen $\longleftarrow$ imRelief `//Previous opening`
4 $\lambda \longleftarrow 0$ `//Current opening size`
5 • ***Main loop*****:**
6 **while** λ<*MaxCriterion* **do**
7 λ=λ+step
`//Current residue calculation ;` γ `opening used for the ultimate opening`
8 imCurrentOpen $\longleftarrow$ γ(imRelief,λ)
9 imCurrentRes $\longleftarrow$ imPreviousOpen − imCurrentOpen
`//Update` $R_\theta(x)$ `and` $q_\theta(x)$ `if imCurrentRes(x) >=` $R_\theta(x)$ `and > 0 (cf. Equation 1)`
10 **foreach** *pixel x* **do**
11 **if** *(imCurrentRes(x)>=*R_θ*(x)) and (imCurrentRes(x)>0)* **then**
`//`$R_\theta(x)$ `and` $q_\theta(x)$ `updating`
12 R_θ(x) $\longleftarrow$ imCurrentRes(x)
13 q_θ(x) $\longleftarrow$ λ
14 imPreviousOpen $\longleftarrow$ imCurrentOpen

Algorithm 1. *Direct implementation of the ultimate opening*

Data: *imRelief* Input image, *imLabels* Labeled Minima of imRelief, *MaxCriterion*(maximum closing step), $\mathcal{N}$ (Neighbourhood)
Result: R_θ (*Transformed* image), q_θ (*Associated* image)

1 • ***Initialisation*** **:**
2 imPropageLabel ← imLabels; imWork ← CANDIDATE ; InitFAH; $R_\theta \leftarrow 0$; $q_\theta \leftarrow 0$
3 **forall** *pixels p de imLabels* **do**
`// Minima neighbor pixels are put in the FAH`
4 Label= imLabels(p)
5 **if** *Label* $\neq$ *0* **then**
6 LakePixelList(Label) ← AddPixel(p)
7 PrevLevel(Label) ← imRelief(p)
8 imWork(p) ← DONE
9 **forall** *pixels* $v \in \{\mathcal{N}_{(p)} \backslash p\}$ **do**
10 **if** *imLabels*$(v) = 0$ **then**
11 AddFAH(p,v,imRelief(v)) ; imWork(v)← QUEUED
12 **for** $i \leftarrow 1$ **to** *NumLabels* **do**
`// Init the criterion measures of each lake`
13 PreviousCriterion[i] ← GetCriterion(LakePixelList[i])
14 Equivalence[i] = i `// No fusion yet: each lake points to itself`
15 • ***End Initialisation*** **:**

Algorithm 2. *Ultimate Criteria Closing Algorithm: Initialization*

```
16 • Propagation:
17 while FAH ≠ EMPTY do
18     (p,v) ← GetFAH
19     if imWork(v) ≠ DONE then
20         Label = imPropagate(p)
21         imPropagate(v) ← Label; imWork(v) ← DONE
22         TopLabel1 = FindTopLabel(Label)
           // LakeGrowing and corresponding Residue update, if required
23         LakeGrowing(prio-ws,TopLabel1,v): Goto 32
24         forall pixels v2 ∈ {N(v)\v} do
25             if imWork(v2) = DONE then
                   // Detect meeting points, and merge lakes if necessary
26                 TopLabel2=FindTopLabel(imPropagate(v2))
27                 if TopLabel1 ≠ TopLabel2 then
                       // meeting point, merge lakes
28                     LakeMerging(prio-ws, TopLabel1, TopLabel2) : Goto 42
29             else
30                 if imWork(v2)=CANDIDATE then
31                     AddFAH(v,v2,imRelief(v2)); imWork(v2)← QUEUED
```

Algorithm 3. *Ultimate Criteria Closing algorithm: Propagation*

LakeGrowing

```
32 LakeGrowing(prio-ws, Label, pixel)
   // MaxCriterion not reached yet.
33 if PreviousCriterion[Label]<MaxCriterion then
34     LakePixelList (Label) ← AddPixel(pixel)
35     Criterion ← GetCriterion(LakePixelList(Label))
       //  Update residue if required (rule 1)
36     if (Criterion>PreviousCriterion[Label] and Criterion<MaxCriterion and
       prio-ws>PrevLevel[Label]) or (Criterion=MaxCriterion) then
37         UpdateResidue(prio-ws,Label,Criterion) : Goto 54
38         PrevLevel[Label] ← prio-ws
39     PreviousCriterion[Label] ← Criterion // rule 2
40 else
       // Lake lab has already reached MaxCriterion. No more residues will be generated on it, but
          flooding process goes on.
41     PrevLevel[Label] ← prio-ws
```

LakeMerging

```
42 LakeMerging(prio-ws,Label1,Label2)
43 Criterion1 ← GetCriterion(LakePixelList(Label1))
44 Criterion2 ← GetCriterion(LakePixelList(Label2))
45 MergedCriterion ← GetCriterion(LakePixelList[Label1 ∪ Label2])
   // Update residues on both lakes, if they verify the constraints
46 if MergedCriterion > Criterion1 and Criterion1+1 <= MaxCriterion and prio-ws >
   PrevLevel[Label1] then
47     UpdateResidue(prio-ws,Label1,Criterion1+1) : Goto 54
48     PrevLevel[Label1] ← prio-ws // lake level is updated only if the residue is updated
49 Repeat lines 46 to 48 for lake 2 (i.e. Label2)
   // Update data structure after merging
50 PrevLevel[Label1] ← ∧(PrevLevel[Label1],PrevLevel[Label2])
51 PreviousCriterion[Label1] ← MergedCriterion // criterion is systematically updated
52 Equivalence[Label2] ← Label1
53 LakePixelList[Label1] ← LakePixelList[Label1] ∪ LakePixelList[Label2]
```

UpdateResidue

```
54 UpdateResidue(prio-ws,Label,Criterion)
55 foreach pixel p in LakePixelList[Label] do
56     residue ← prio-ws - imRelief(p) // Current residue in p
57     imRelief(p) ← prio-ws // flood level at last residue updating
58     if (residue ≥ R_θ(p) and residue > 0) then
59         R_θ(p) ← residue;q_θ(p) ← Criterion
```