

Multi-Scale Component-Tree: A Hierarchical Representation for Sparse Objects

Romain PERRIN, Aurélie LEBORGNE, Nicolas PASSAT,
Benoît NAEGEL, and Cédric WEMMERT

Firenze, 15-18 may 2024

DGMM2024

Summary

I. Introduction

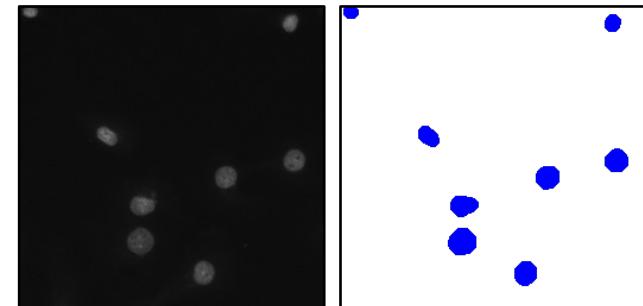
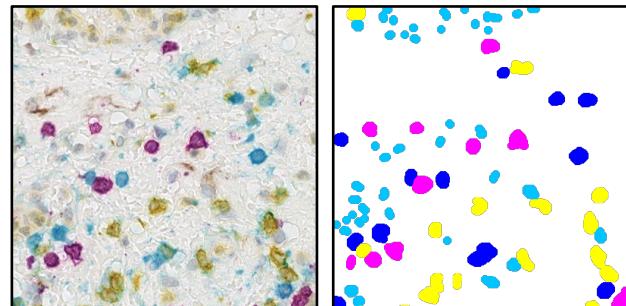
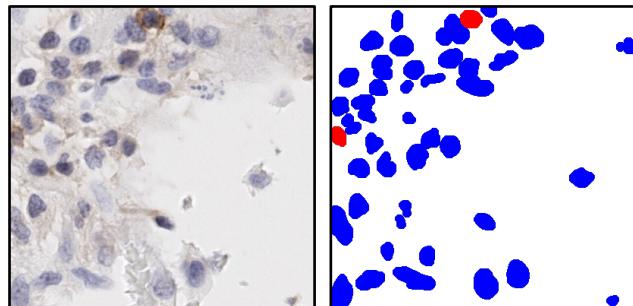
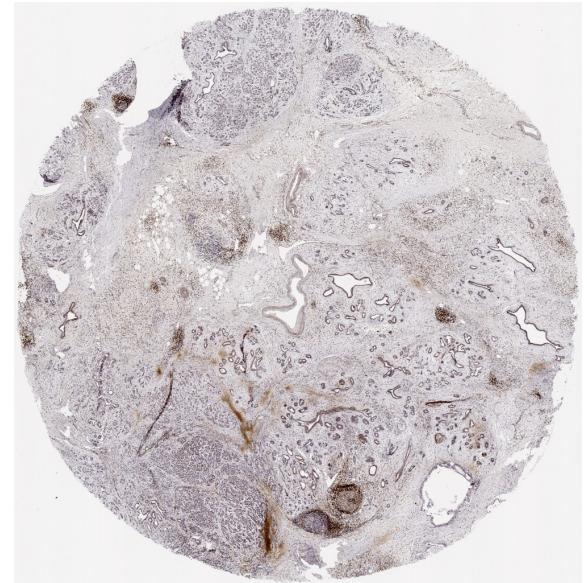
II. Building a Multi-Scale Component-Tree

III. MSCT: applications

IV. Conclusion

I – Introduction: images

- Biological images (IHC, IF, mIHC, mIF...)
 - Large images of tissue samples
 - Sparse objects of interest (cells/nuclei)
 - Background mostly irrelevant
 - Pixel-based representation
 - Could benefit from a more efficient representation

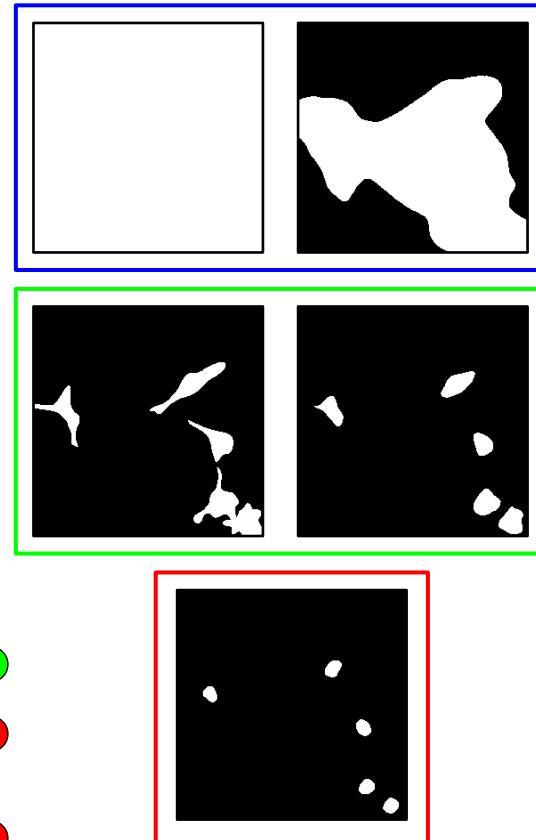
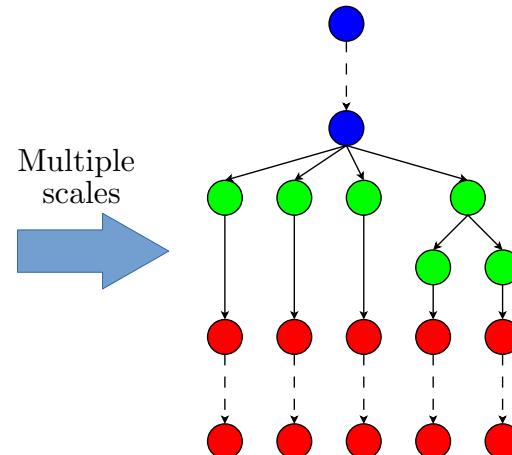
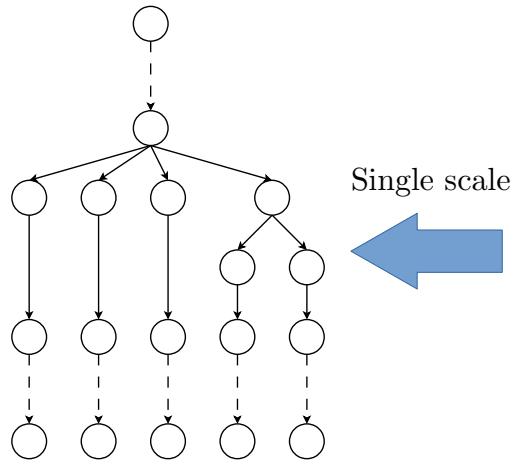


I – Introduction: image representations

- **Pixel-based**
- **Block-based**
 - quadtree, octree...
- **Frequency domain**
 - Fourier transform, wavelets, ridgelets, contourlets...
- **Region-based**
 - superpixels, normalized cuts, Felzenszwalb & Huttenlocher, watershed...
- **Hierarchical**
 - Binary partition tree, component-tree, tree of shapes, α -tree, (ω) -tree...

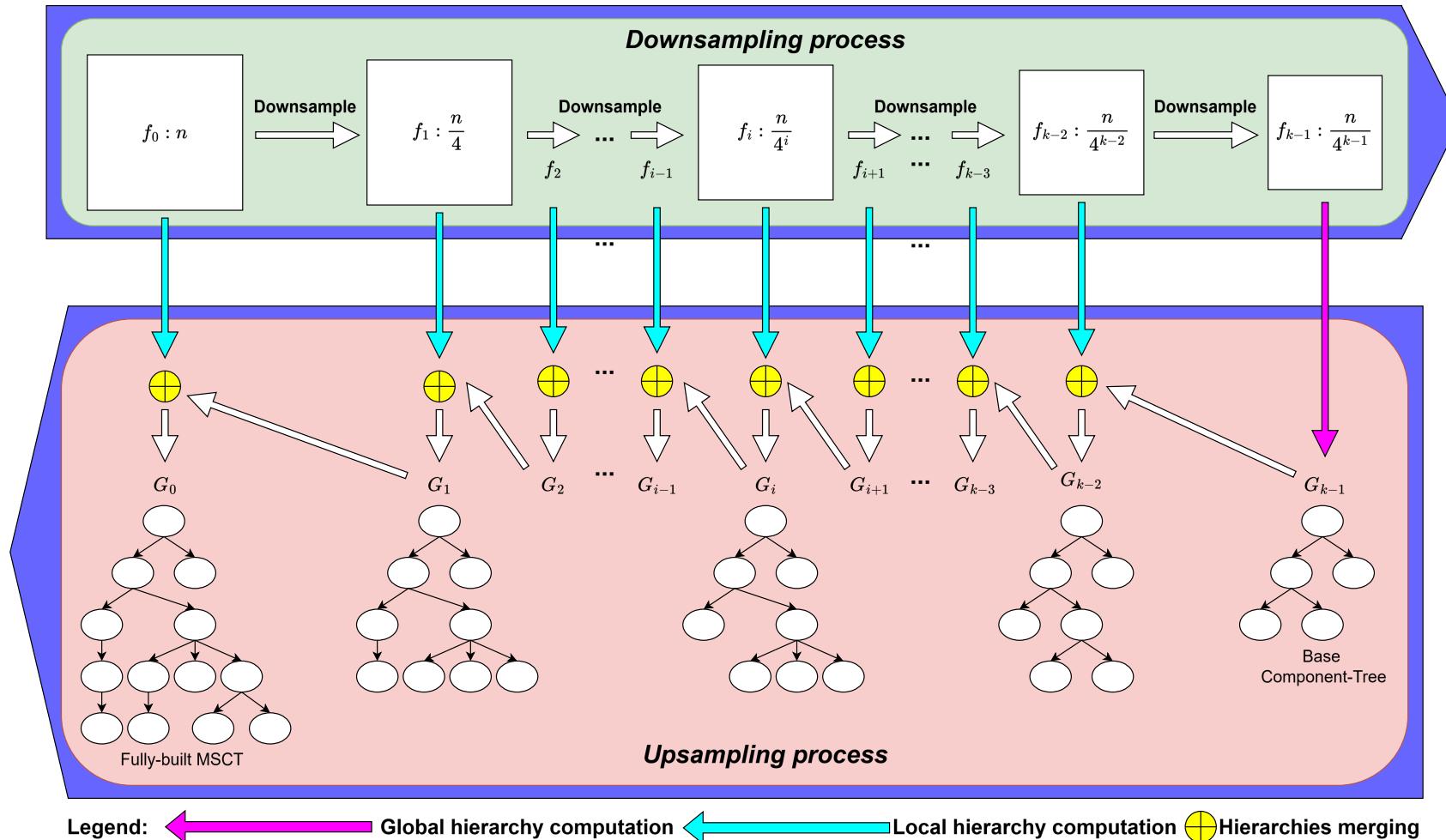
I – Introduction: purpose

- Using the component-tree [1] as a basis
- Using different scales of the image
- Promote nodes with objects of interest to higher scales
- Keep other less relevant nodes at low scales



[1] P. Salembier, A. Oliveras, L. Garrido, Anti-extensive connected operators for image and sequence processing, IEEE Transactions on Image Processing, vol. 7, pp. 555–570, 1998.

II – The MSCT: core principles



II – The MSCT: construction algorithm

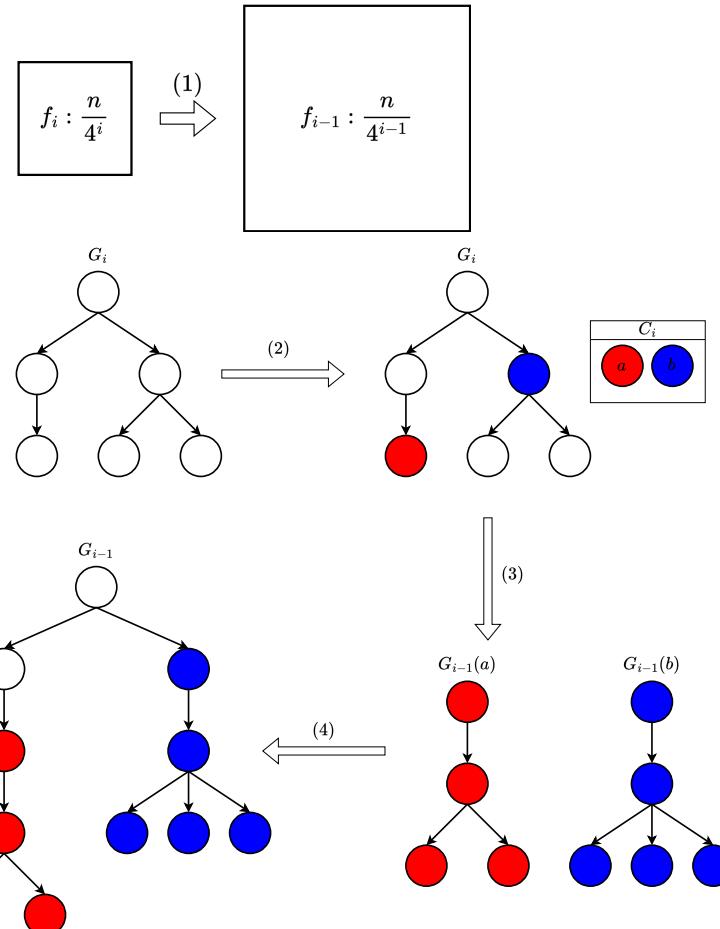
Algorithm 1: MSCT construction.

Data: $f : \mathbb{S}_n \rightarrow \mathbb{V}$ (gray-scale image), $k \in \mathbb{N}^*$ number of scales
Result: $G = (V, E)$ Multi-Scale Component-Tree of f

```
1 begin
2   |    $F \leftarrow \{f_0, \dots, f_{k-1}\}$  with  $f_0 = f$  and
3   |   |    $\forall i \in \llbracket 1, k-1 \rrbracket, f_i \leftarrow \text{Downsample}(f_{i-1})$ 
4   |    $G \leftarrow G_{k-1} \leftarrow \text{ComputeGlobalHierarchy}(f_{k-1})$ 
5   |   for  $i$  from  $k-2$  down to 0 do
6   |   |    $C_i \leftarrow \text{SelectNodes}(G, i)$ 
7   |   |   foreach  $N \in C_i$  do
8   |   |   |    $G_i(N) \leftarrow \text{ComputeLocalHierarchy}(f_i, N)$ 
9   |   |   |   MergeHierarchies( $G, G_i(N), N$ )
10  |   |   end
11 end
```

II – The MSCT: operations

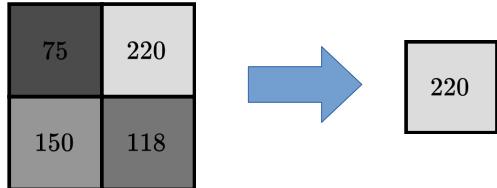
- Downsampling: $f_i \rightarrow f_{i+1}$
 - Downsampling operation (1)



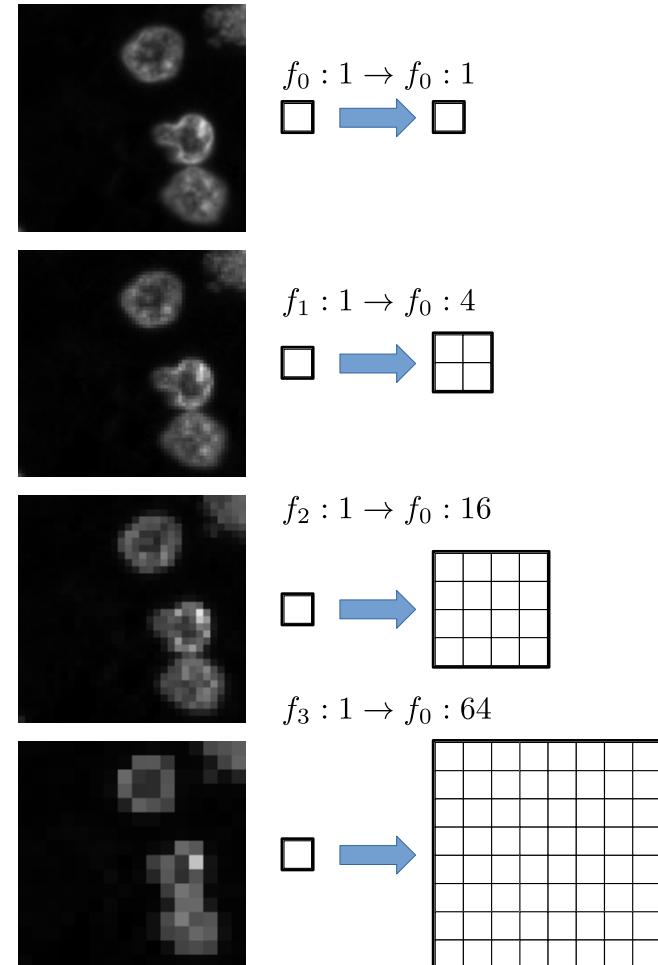
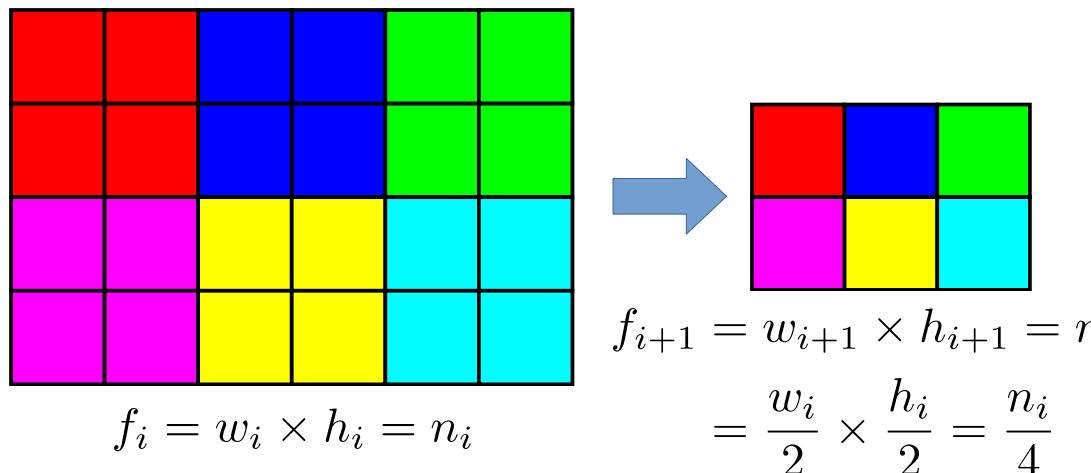
- Upsampling: $G_i \rightarrow G_{i-1}$
 - Computing a global hierarchy G_{k-1}
 - Selecting C_i nodes from G_i (2)
 - Computing local hierarchies of C_i on f_{i-1} (3)
 - Enriching G_i to create G_{i-1} (4)

II – Building the MSCT: downsampling

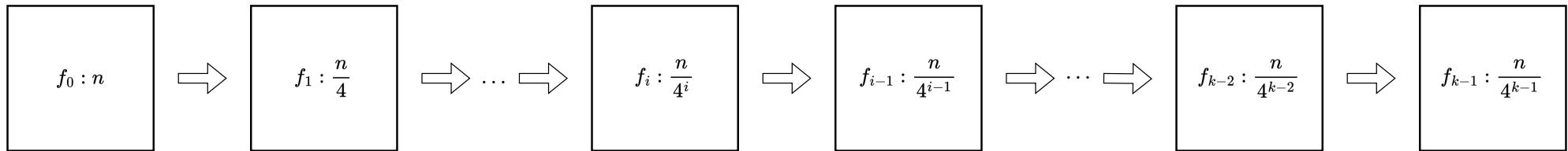
- Downsampling: promote flat zones of high contrast
- Sliding window with a maximum strategy



- \Rightarrow MaxPooling of size 2, stride 2

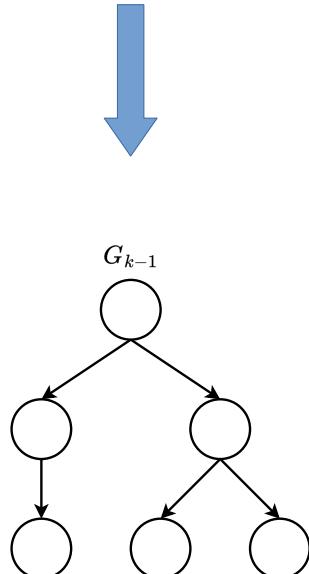


II – Building the MSCT: base component-tree



- Base component-tree (single scale global hierarchy)

- Built on f_{k-1}
- Optimal algorithm in quasi-linear time [2]
- $f_{k-1} : \frac{n}{4^{k-1}}$
- Cost of computing $G_{k-1} = (V_{k-1}, E_{k-1}) : \mathcal{O}\left(\frac{n}{4^{k-1}} \log\left(\frac{n}{4^{k-1}}\right)\right)$
- Najman & Couprie's algorithm [3]

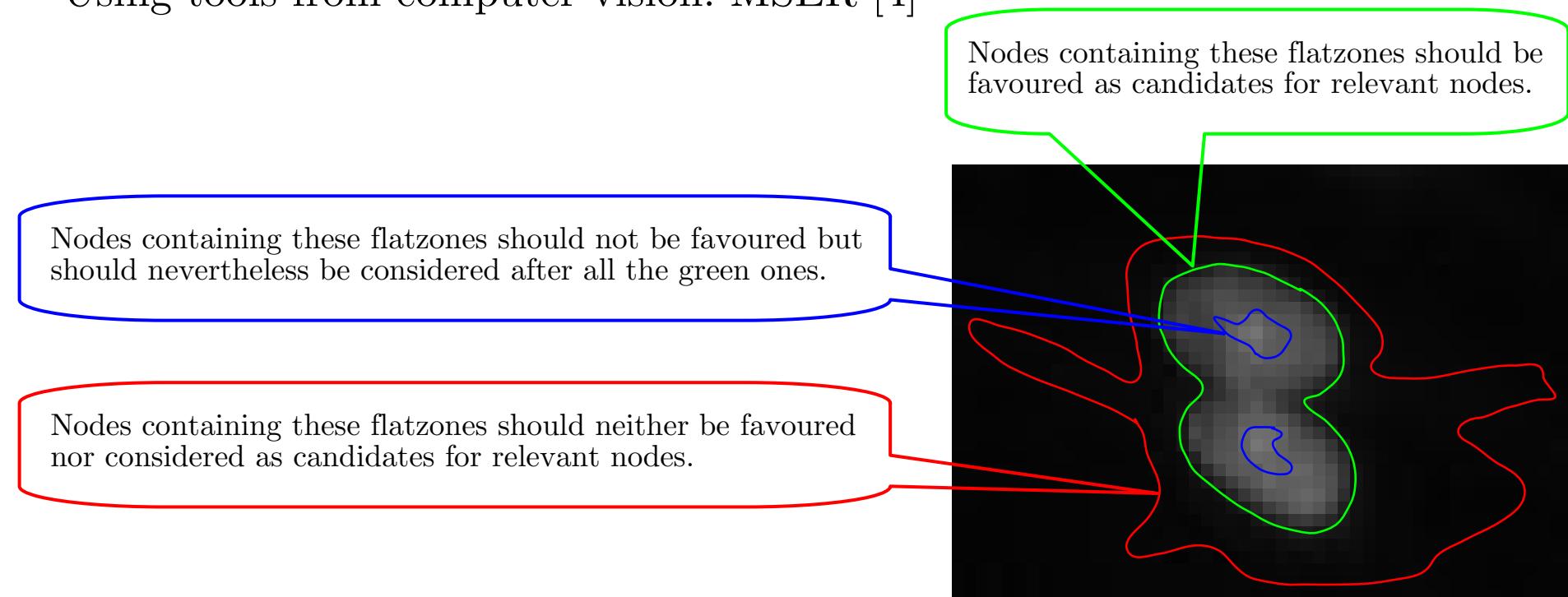


[2] E. Carlinet, T. Géraud, A comparative review of component tree computation algorithms, IEEE Transactions on Image Processing, vol. 23, pp. 3885–3895, 2014.

[3] L. Najman., M. Couprie, Building the component tree in quasi-linear time, IEEE Transactions on Image Processing, vol. 15, pp. 3531–3539, 2006.

II – Building the MSCT: nodes metric

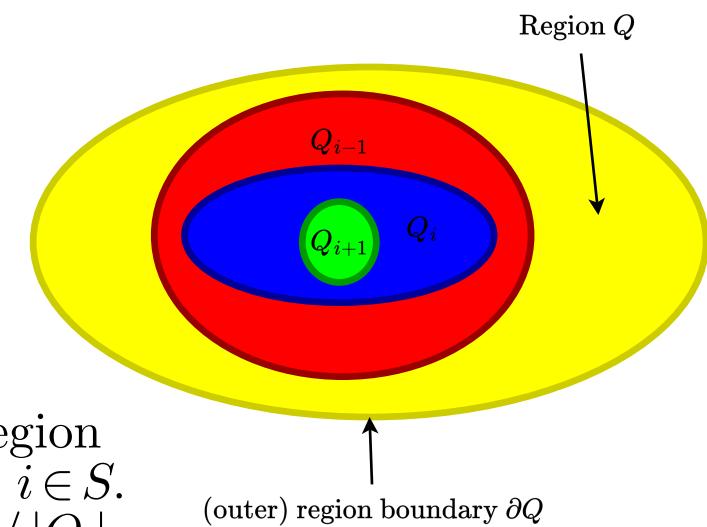
- A metric needs to be defined in order to select relevant nodes
- Using tools from computer vision: MSER [4]



[4] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, vol. 22, pp. 761–767, 2004.

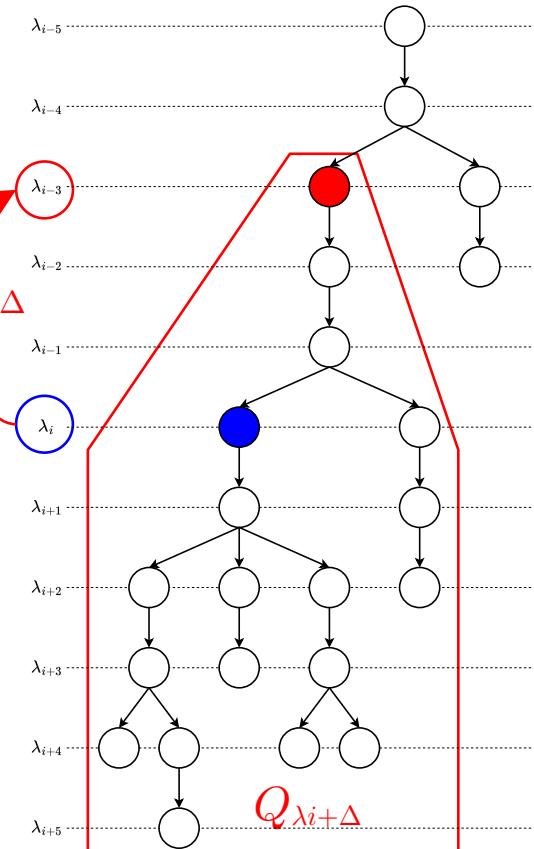
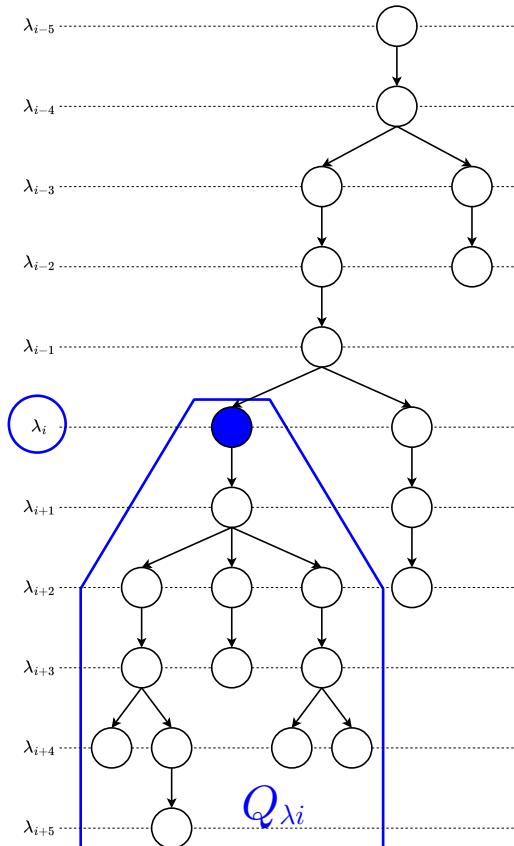
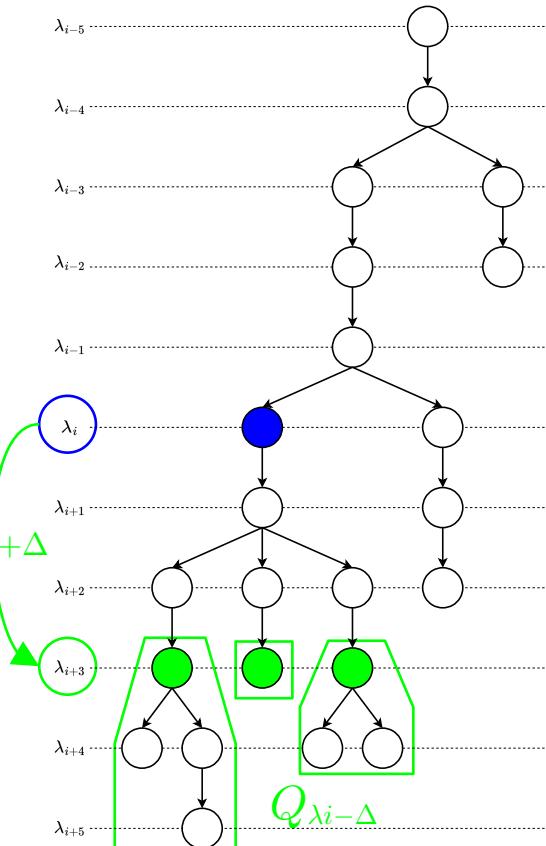
II – Building the MSCT: MSER in theory

- **Image I** is a mapping $I: D \subset \mathbb{Z}^2 \rightarrow S$
 - S is totally ordered (total, antisymmetric and transitive relation \leq exist)
 - An adjacency relation $A \subset D \times D$ exists denoting two adjacent points by pAq .
- **Region Q** is a connected subset of D .
For each $p, q \in Q$, there is a sequence $p, a_1, a_2, \dots, a_n, q$ such as $pAa_1, a_1Aa_2, \dots, a_{n-1}Aa_n, a_nAq$.
- **Outer region boundary:** $\partial Q = \{ q \in D \setminus Q, \exists p \in Q, qAp \}$.
- **Extremal region:** $Q \subset D$ is a region such that for all $p \in Q, q \in \partial Q, I(p) > I(q)$ (maximum intensity region)
- **Maximally stable extremal region:** Let Q_i an extremal region such that all points on it have an intensity smaller than $i \in S$. Extremal region Q_{i*} is maximally stable iff $|Q_{i+\Delta} \setminus Q_{i-\Delta}| / |Q_i|$ has a local minimum at $i*$.



II – Building the MSCT: MSER in practice

- MSER on a component-tree: using the flatzones to compute $|Q_{i+\Delta} \setminus Q_{i-\Delta}| / |Q_i|$



II – Building the MSCT: selecting relevant nodes (algorithm)

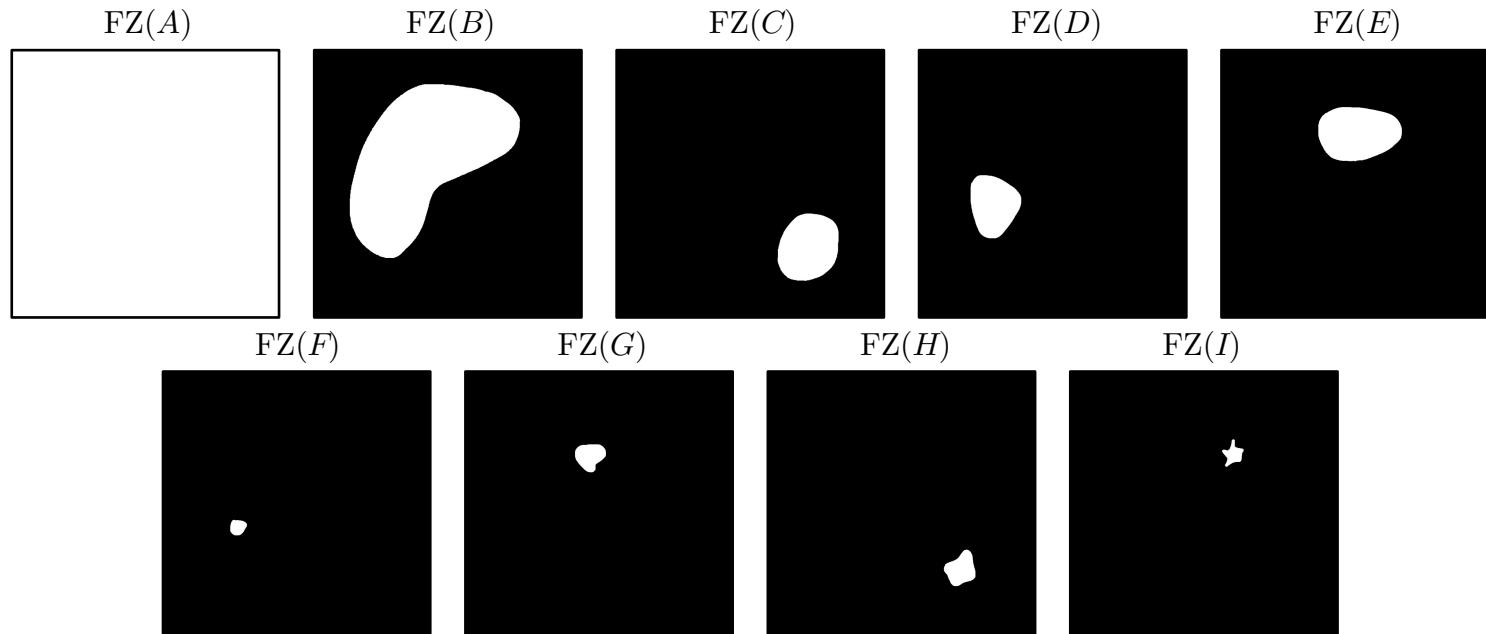
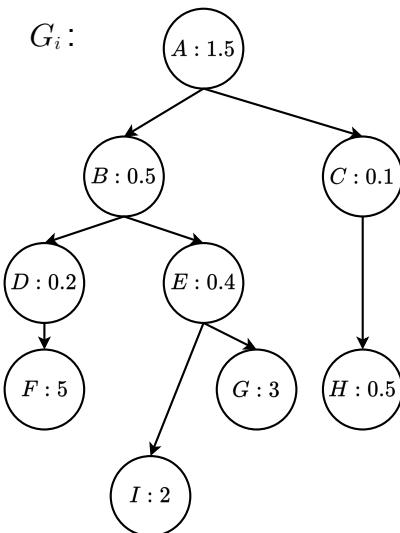
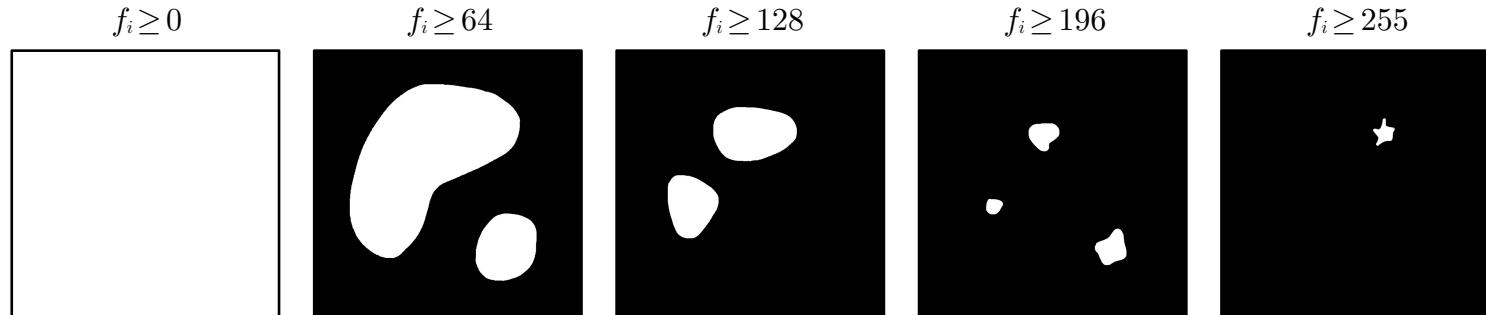
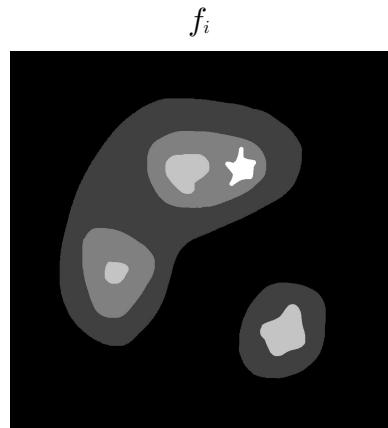
Algorithm 1: Select nodes for upsampling.

Data: $G_i = (V_i, E_i)$ Multi-Scale Component-Tree of f_i

Result: $C_i \subset V_i$ Roots of disjoint subtrees of G

```
1 begin
2      $C_i \leftarrow \emptyset$ 
3      $Q \leftarrow \{v \in V_i \mid \forall a, b \in V_i, \text{MSER}(a) \leq \text{MSER}(b)\}$ 
4     while  $|Q| > 0$  do
5          $v \leftarrow \text{PopFirst}(Q)$ 
6          $C_i \leftarrow C_i \cup \{v\}$ 
7          $A_v \leftarrow \text{Ancestors}(G_i, v)$ 
8          $D_v \leftarrow \text{Descendants}(G_i, v)$ 
9          $Q \leftarrow Q \setminus \{A_v \cup D_v\}$ 
10    end
11 end
```

II – Building the MSCT: selecting relevant nodes (example)

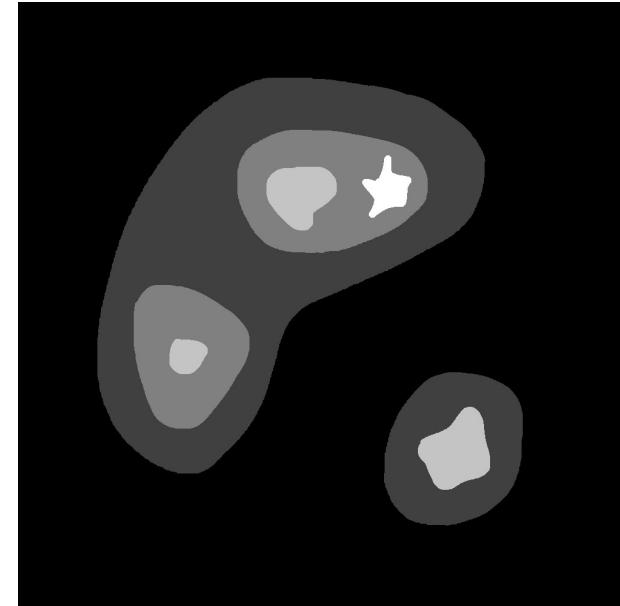
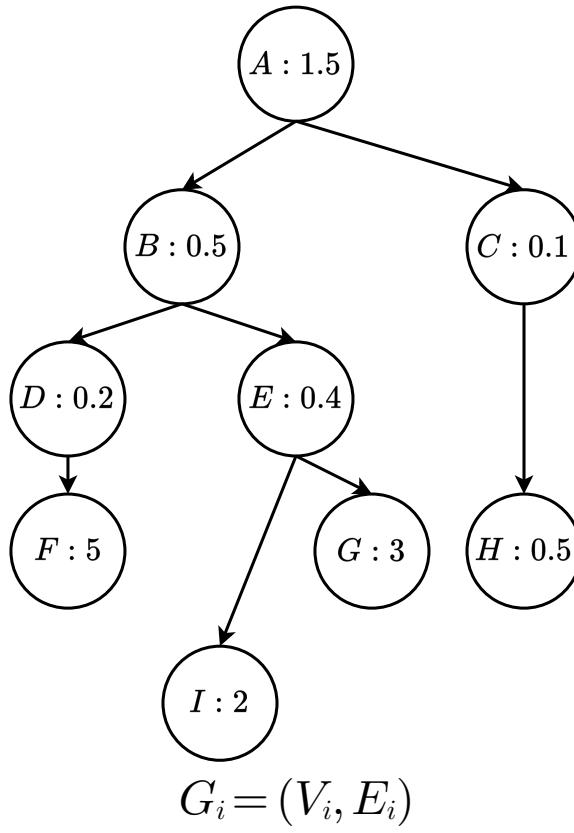


II – Building the MSCT: selecting relevant nodes (example)

Input: $V_i = \{A, B, C, D, E, F, G, H, I\}$

Initialization

$Q = \{C, D, B, E, H, A, I, G, F\}$
 $C_i = \emptyset$



$$f_i : w_i \times h_i = n_i$$

II – Building the MSCT: selecting relevant nodes (example)

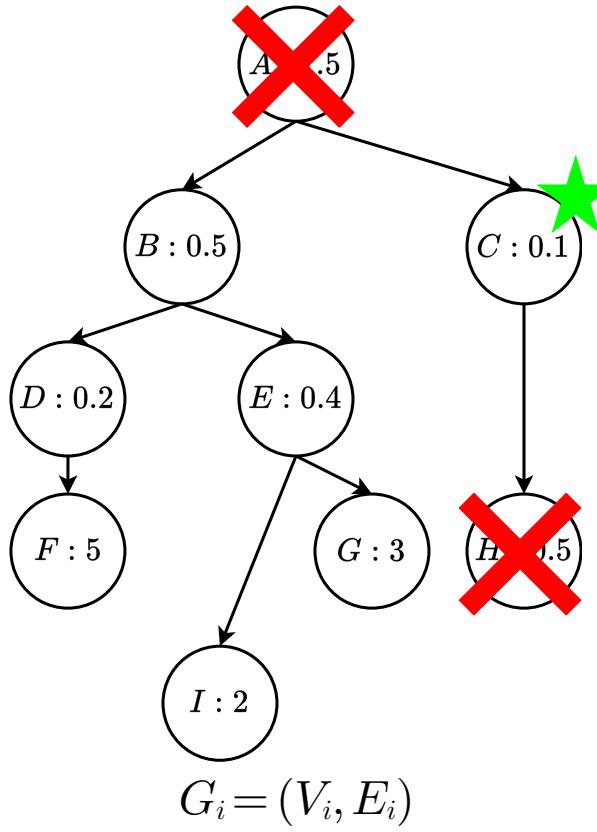
Input: $V_i = \{A, B, C, D, E, F, G, H, I\}$

Initialization

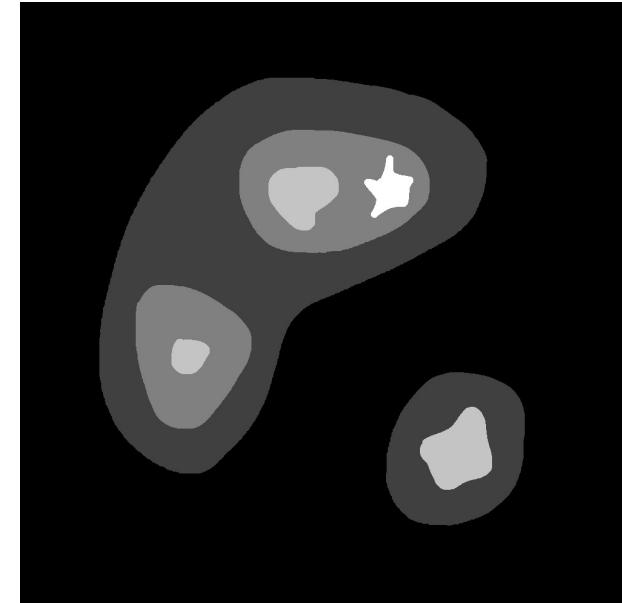
$Q = \{C, D, B, E, H, A, I, G, F\}$
 $C_i = \emptyset$

Iteration 1

$Q = \{C, D, B, E, H, A, I, G, F\}$
 $C_i = \{C\}$



$$f_i : w_i \times h_i = n_i$$



II – Building the MSCT: selecting relevant nodes (example)

Input: $V_i = \{A, B, C, D, E, F, G, H, I\}$

Initialization

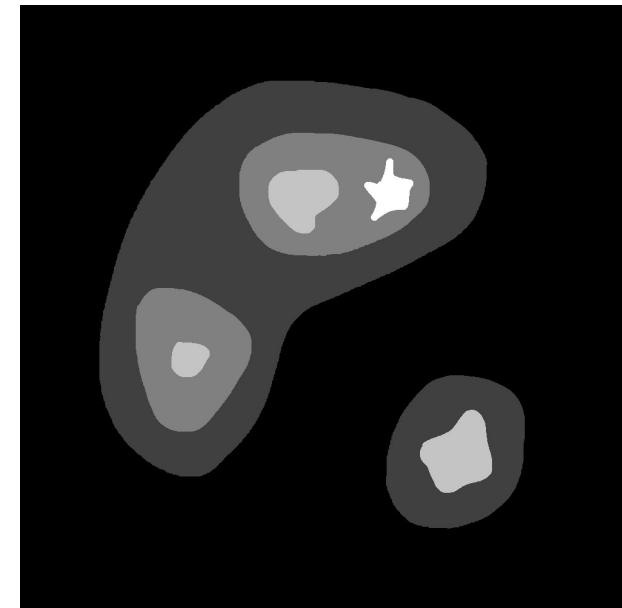
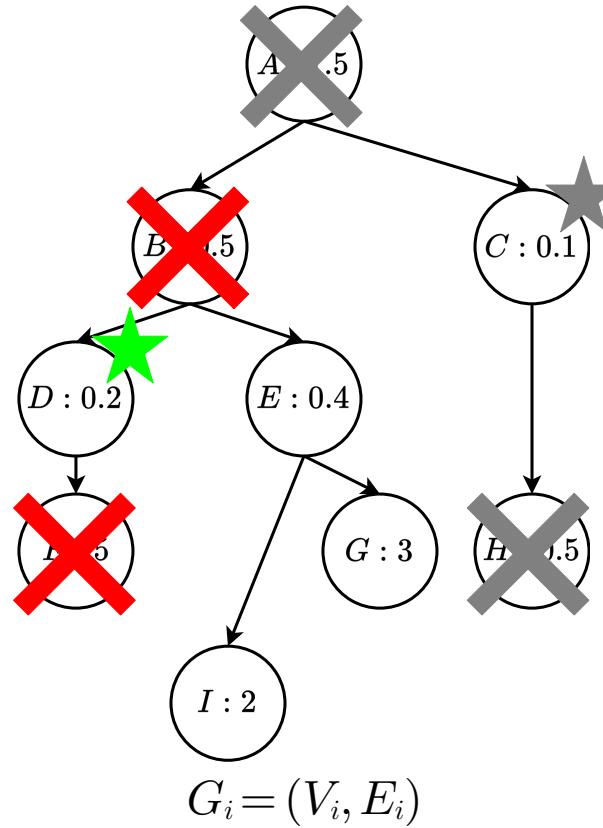
$Q = \{C, D, B, E, H, A, I, G, F\}$
 $C_i = \emptyset$

Iteration 1

$Q = \{C, D, B, E, H, A, I, G, F\}$
 $C_i = \{C\}$

Iteration 2

$Q = \{D, B, E, I, G, F\}$
 $C_i = \{C, D\}$



$$f_i : w_i \times h_i = n_i$$

II – Building the MSCT: selecting relevant nodes (example)

Input: $V_i = \{A, B, C, D, E, F, G, H, I\}$

Initialization

$Q = \{C, D, B, E, H, A, I, G, F\}$
 $C_i = \emptyset$

Iteration 1

$Q = \{C, D, B, E, H, A, I, G, F\}$
 $C_i = \{C\}$

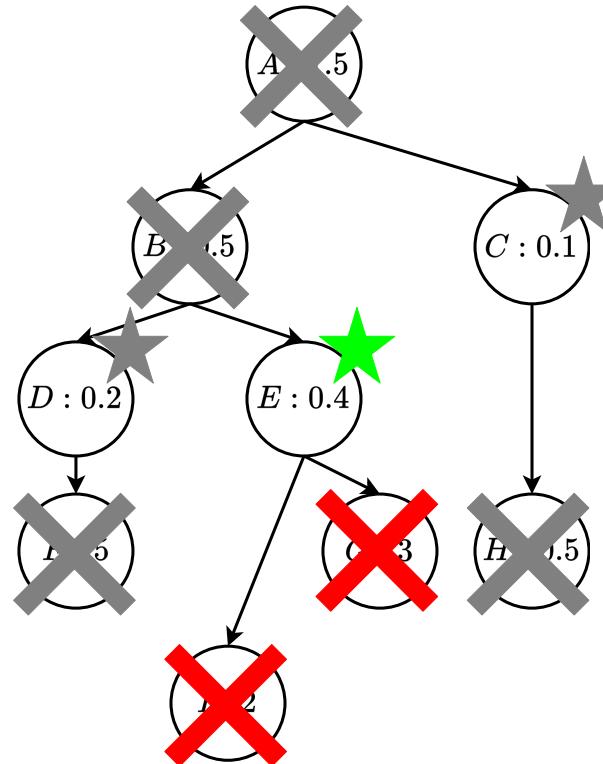
Iteration 2

$Q = \{D, B, E, I, G, F\}$
 $C_i = \{C, D\}$

Iteration 3

$Q = \{E, I, G\}$
 $C_i = \{C, D, E\}$

Output: $C_i = \{C, D, E\}$



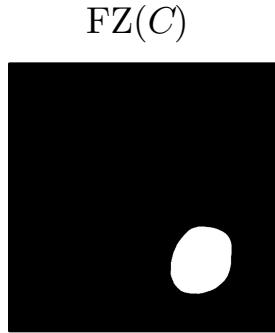
$$f_i : w_i \times h_i = n_i$$



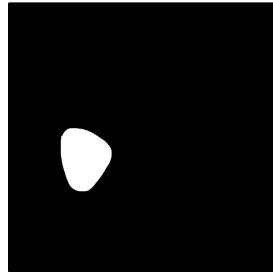
II – Building the MSCT: selecting relevant nodes (example)

Input: $V_i = \{A, B, C, D, E, F, G, H, I\}$

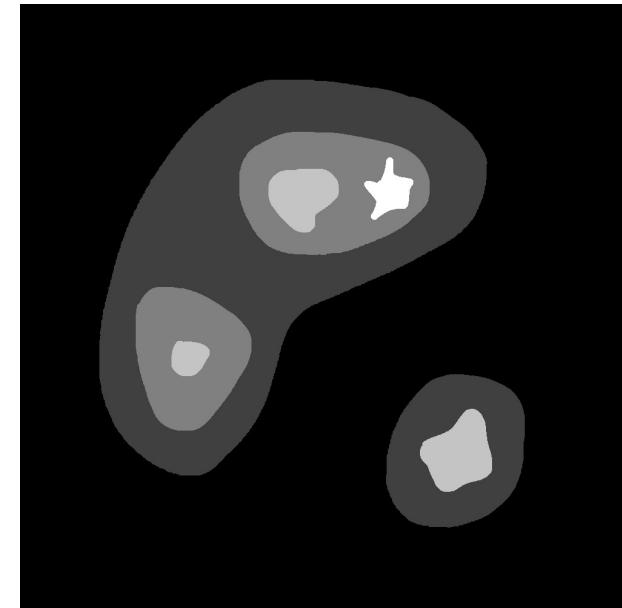
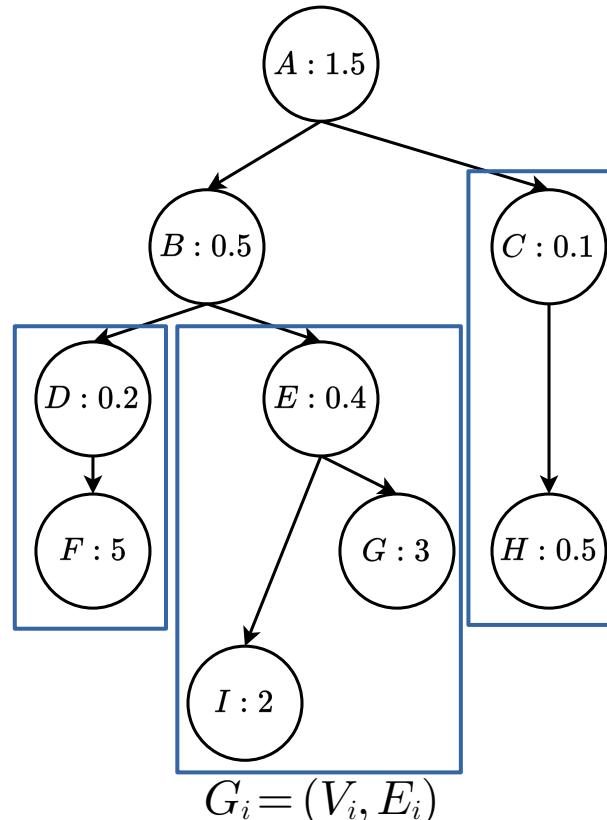
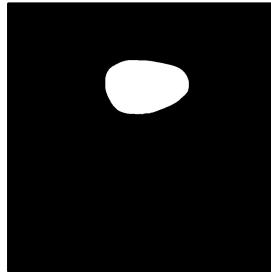
Output: $C_i = \{C, D, E\}$



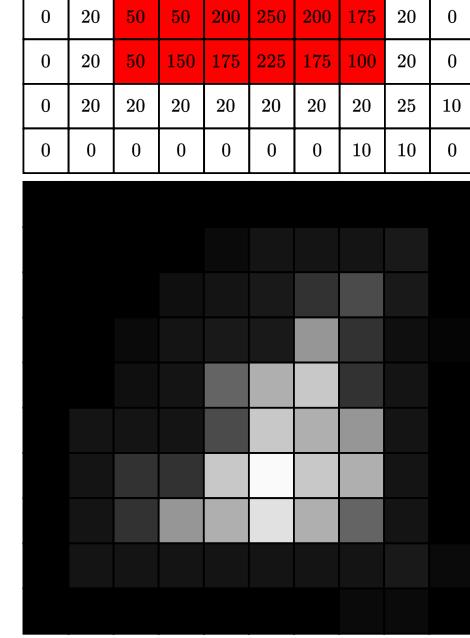
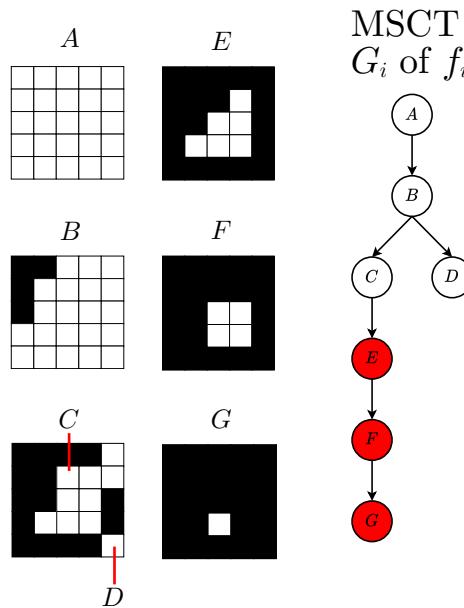
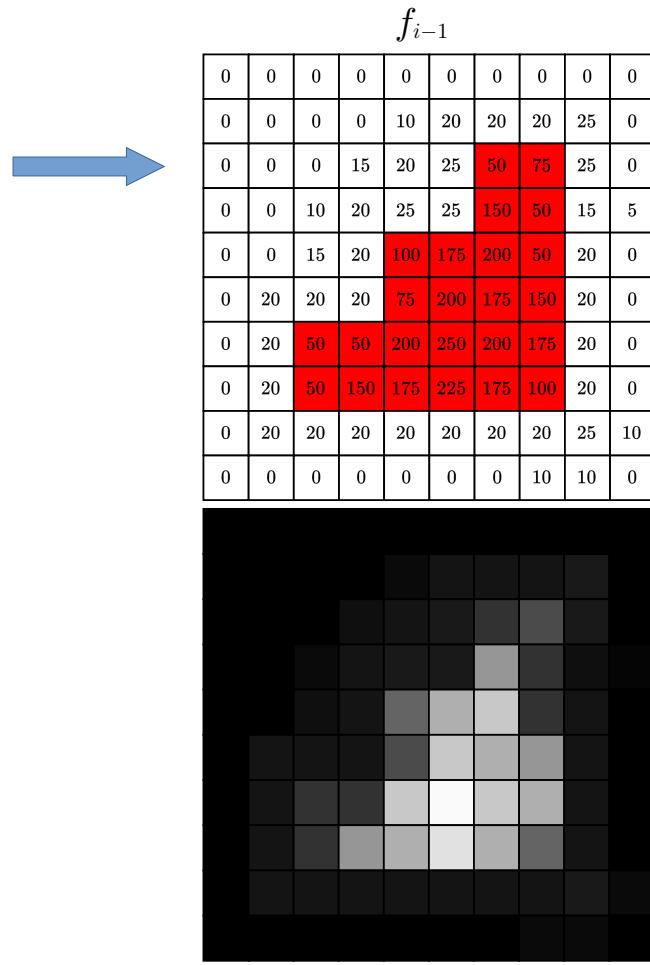
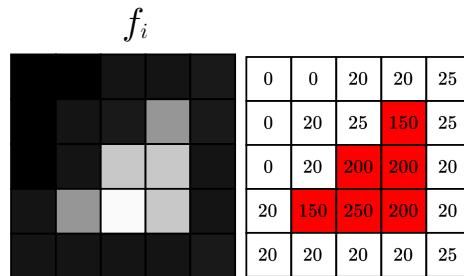
FZ(D)



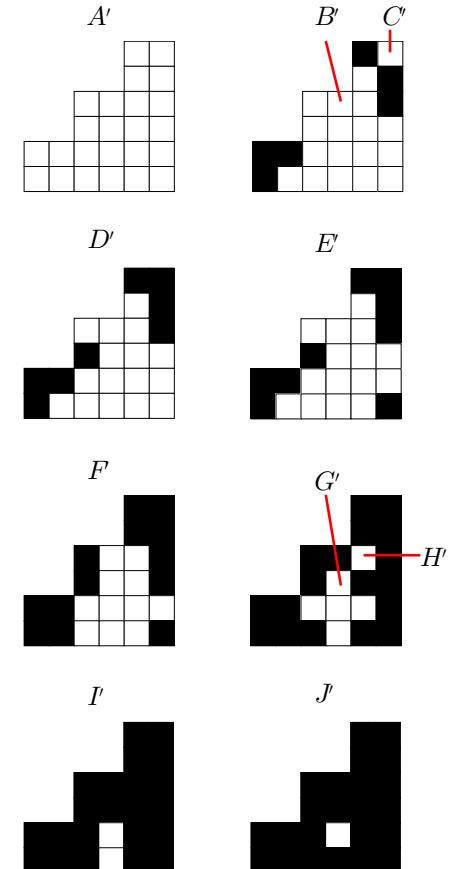
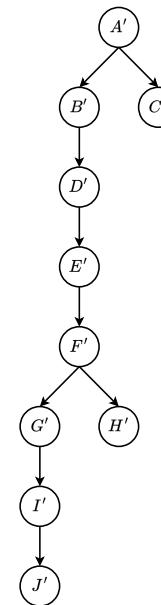
FZ(E)



II – Building the MSCT: computing local hierarchies

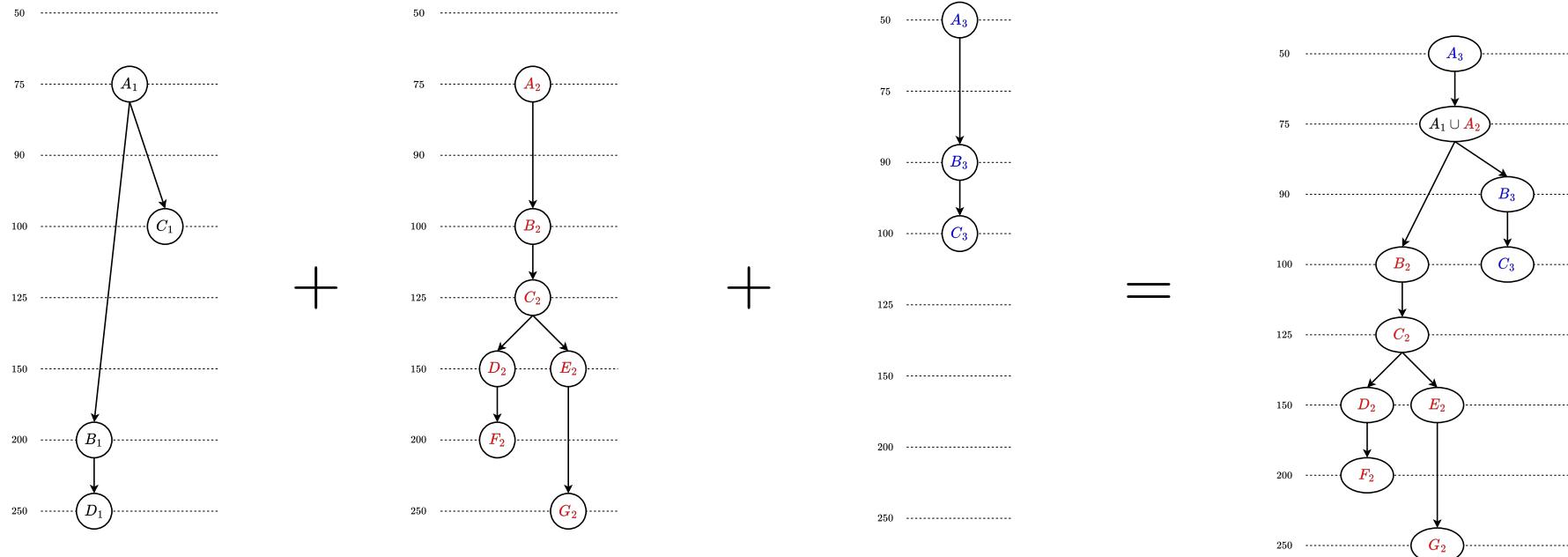


Partial component-tree
 $G_{i-1}(E)$ on $f_{i-1} \cap \text{FZ}(E)$



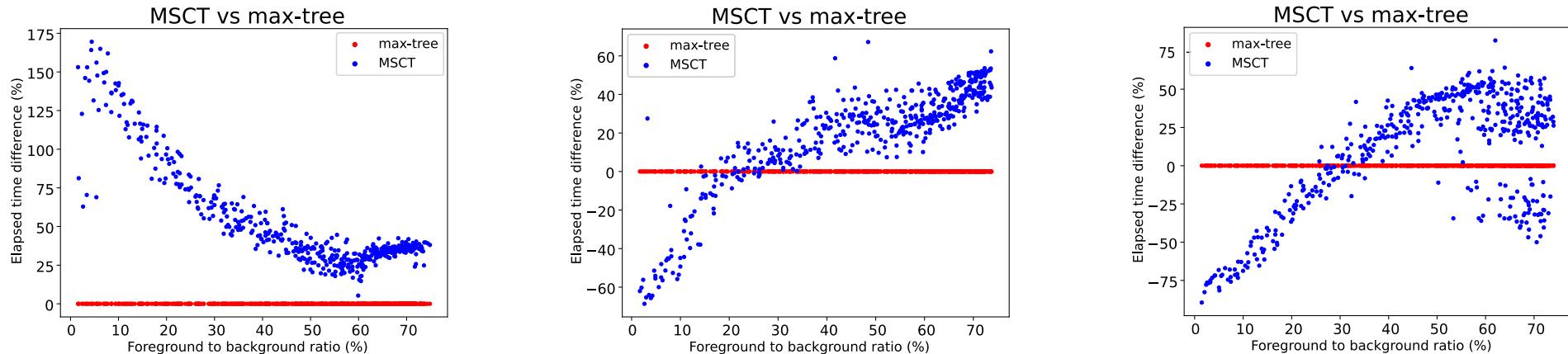
II – Building the MSCT: merging hierarchies

- Merging may introduce new nodes/gray-levels
- Flat zones may now represent unions of flat zones
- No redundancy concerning pixels

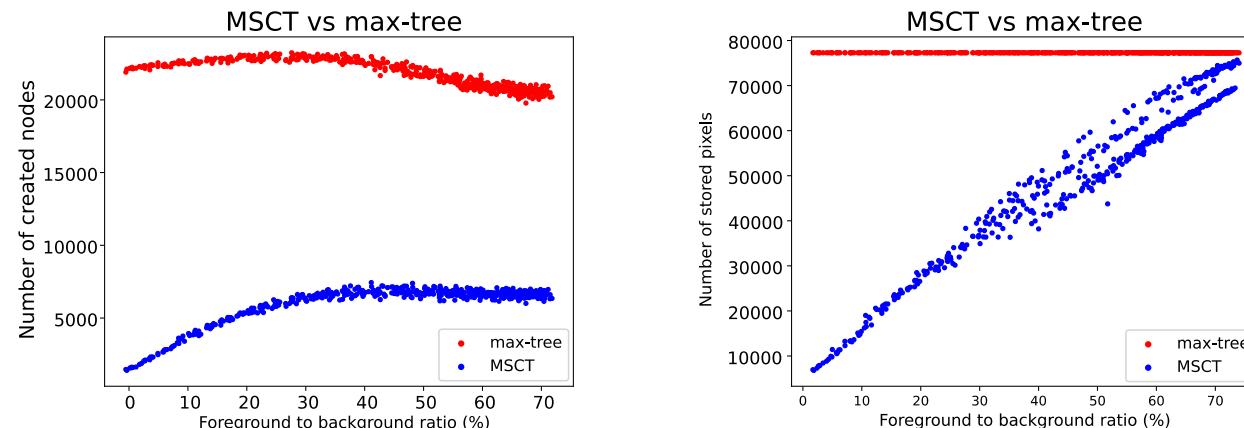


II – Building the MSCT: Performances

- Time computation measurements, $k=\{2,3,4\}$



- Memory usage measurements, $k=3$



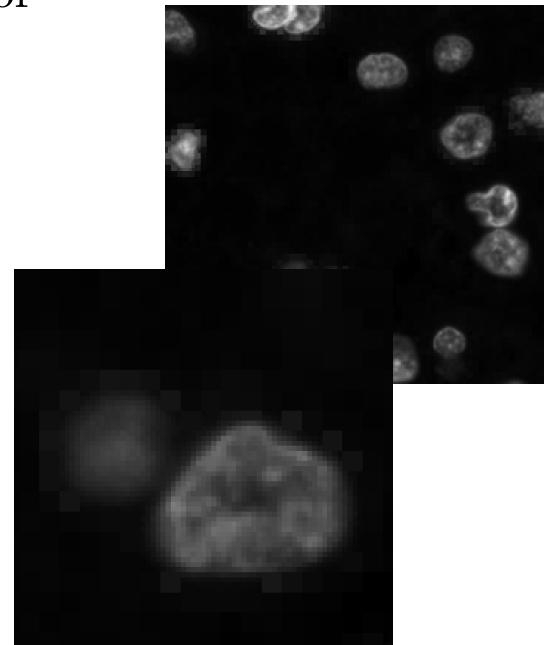
III – Using the MSCT: segmentation (algorithm)

Algorithm 1: MSCT segmentation

Data: $G = (V, E)$ MSCT, $MSER_{max} \in \mathbb{N}$ maximum MSER value,
 $f : \mathbb{Z}^2 \rightarrow \mathbb{N}$ input image, $n \in \mathbb{N}$ subdivision factor

Result: $P = \{P_0, \dots, P_k\}$

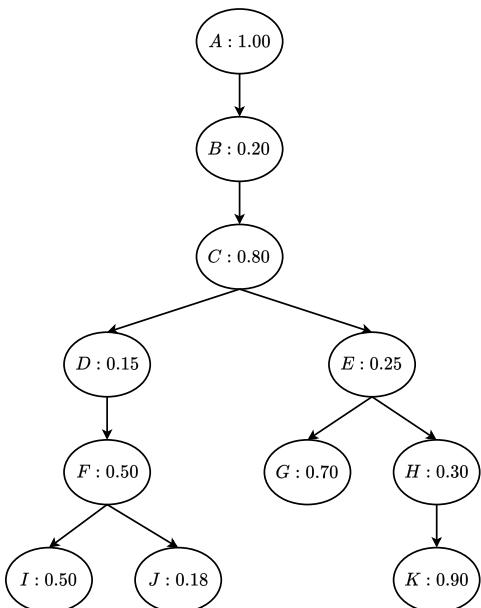
```
1 begin
2   |    $P \leftarrow \emptyset$ 
3   |    $C_0 \leftarrow \text{SelectNodes}(G)$ 
4   |    $C'_0 \leftarrow \text{FilterTree}(G, C_0, MSER_{max})$ 
5   |   for  $c \in C'_0$  do
6     |     |    $c' \leftarrow \text{FillHoles}(\text{Otsu}(\text{GaussianFilter}(c)))$ 
7     |     |    $S \leftarrow \text{Watershed}(f, c, \text{UltimateErosion}(c'))$ 
8     |     |    $P \leftarrow P \cup S$ 
9   |   end
10  end
```



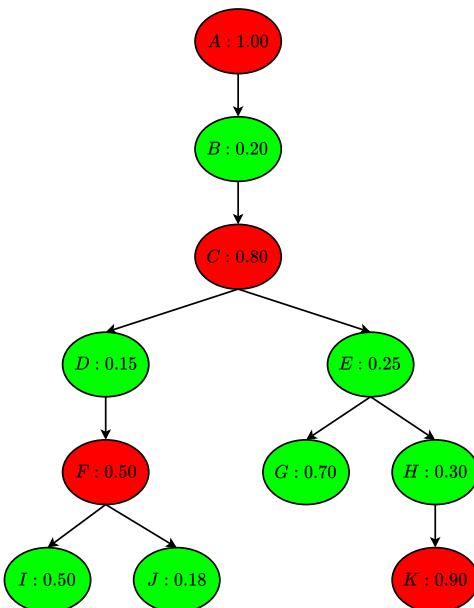
III – Using the MSCT: segmentation (example)

$$KeepNode(n, v) = \begin{cases} True, & \text{if } n.\text{nChildren} = 0 \wedge n.mser \leq v \\ True, & \text{if } n.\text{nChildren} > 0 \wedge n.mser \leq v \wedge \forall n' \in n.\text{children}, n'.mser \leq v \\ False, & \text{otherwise} \end{cases}$$

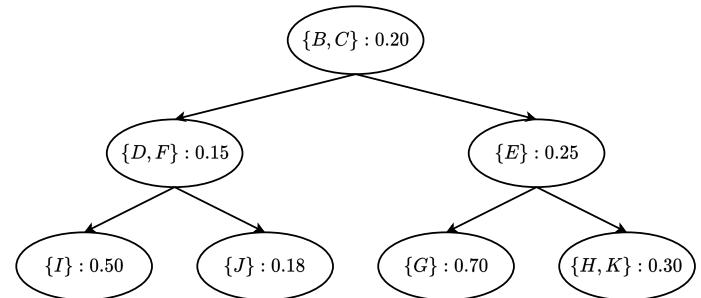
MSCT with MSER values



MSCT with MSER local branch minima

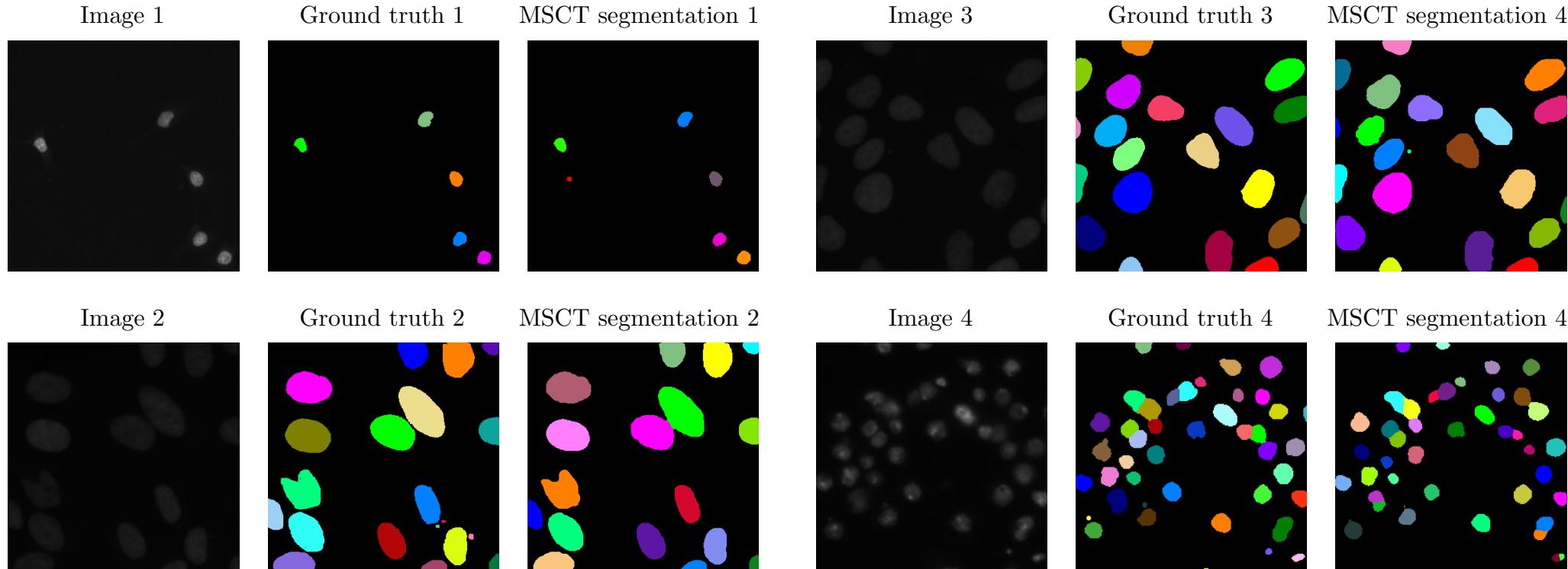


Simplified MSCT with each branch collapsed to its local MSER minima



III – Using the MSCT: segmentation (results)

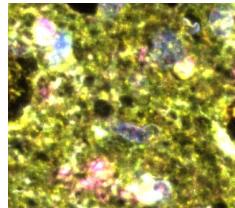
- MSCT used to segment images from kaggle's 2018 Data Science Bowl [5]
- Intersection over union: IoU = 0.465



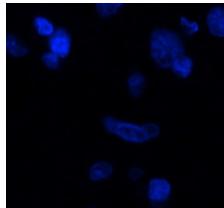
[5] <https://www.kaggle.com/c/data-science-bowl-2018/>

III – Using the MSCT: segmentation & classification

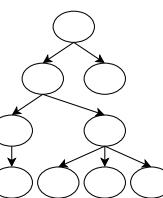
Multiplex image



DAPI channel



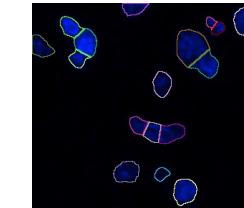
MSCT



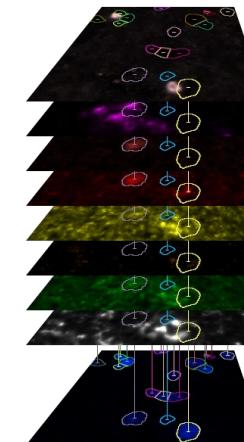
Segmentation



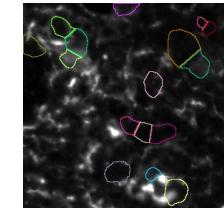
DAPI contours



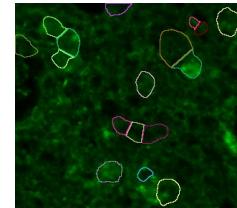
Projection



GFAP channel



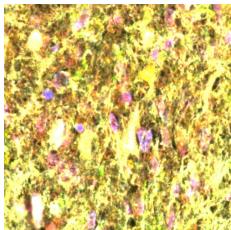
MS4A4A channel



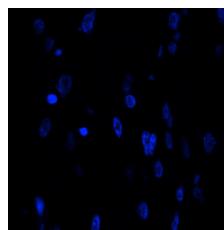
$$\text{Feature vector : } \mathbf{c}_i = \left[\sum_p f^j(p) \mid p \in n_i \right]_{j=1}^k$$

With $f^j(p)$ the value of pixel p on the j^{th} channel of the multiplex image f containing k channels.

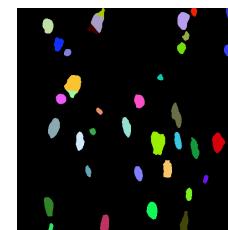
Multiplex 1



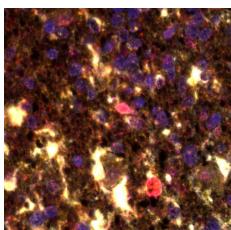
DAPI channel 1



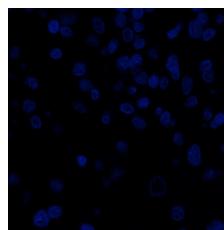
Segmentation 1



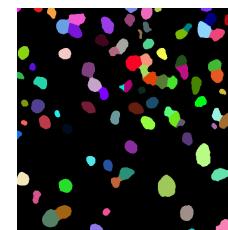
Multiplex 2



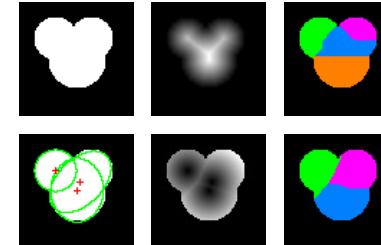
DAPI channel 2



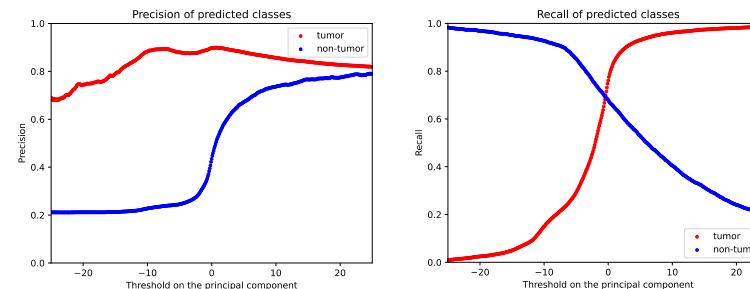
Segmentation 2



Watershed from EDT vs watershed from fake EDT with DTECMA [6]

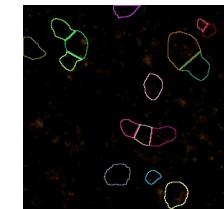


Precision and recall of binary classification (tumor/non tumor)

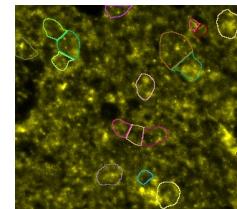


[6] T. Zou, T. Pan, M. Taylor, H. Stern, Recognition of overlapping elliptical objects in a binary image, Pattern Analysis and Applications, vol. 24, pp. 1193-1206, 2021.

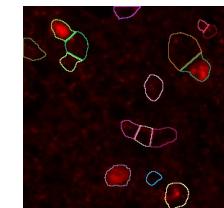
CD3 channel



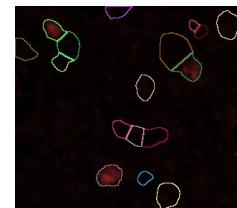
IDH1 channel



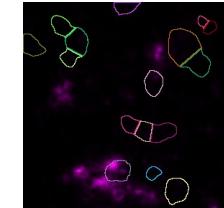
CD206 channel



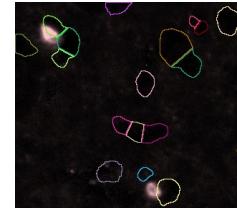
ATRX channel



CD68 channel



CD34 channel



IV – Conclusion

- Python code is available [7]
- Extension to dark on bright images (min-tree)
- Extension to colour images (MSCG for component-graphs [8])
- Generalization to other hierarchical structure
 - MSToS for the Tree of Shapes [9]
 - MS α -T for the α -tree or MS(ω)-T for the (ω) -tree [10]
- Use of pyramidal decomposition (tiff)

[7] <https://github.com/Romain96/MSCT>

[8] N. Passat, B. Naegel, Component-Trees and Multivalued Images: Structural Properties, In : Journal of Mathematical Imaging and Vision, vol. 49, issue 1, pp. 37-50, 2014.

[9] P. Monasse, F. Guichard, Fast computation of a contrast invariant image representation. IEEE Transactions on Image Processing, vol. 9, issue 5, pp. 860–872, 2000.

[10] P. Soille, Constrained connectivity for hierarchical image partitioning and simplification, In : IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 30, issue 7, pp. 1132-1145, 2008.

Thank you !

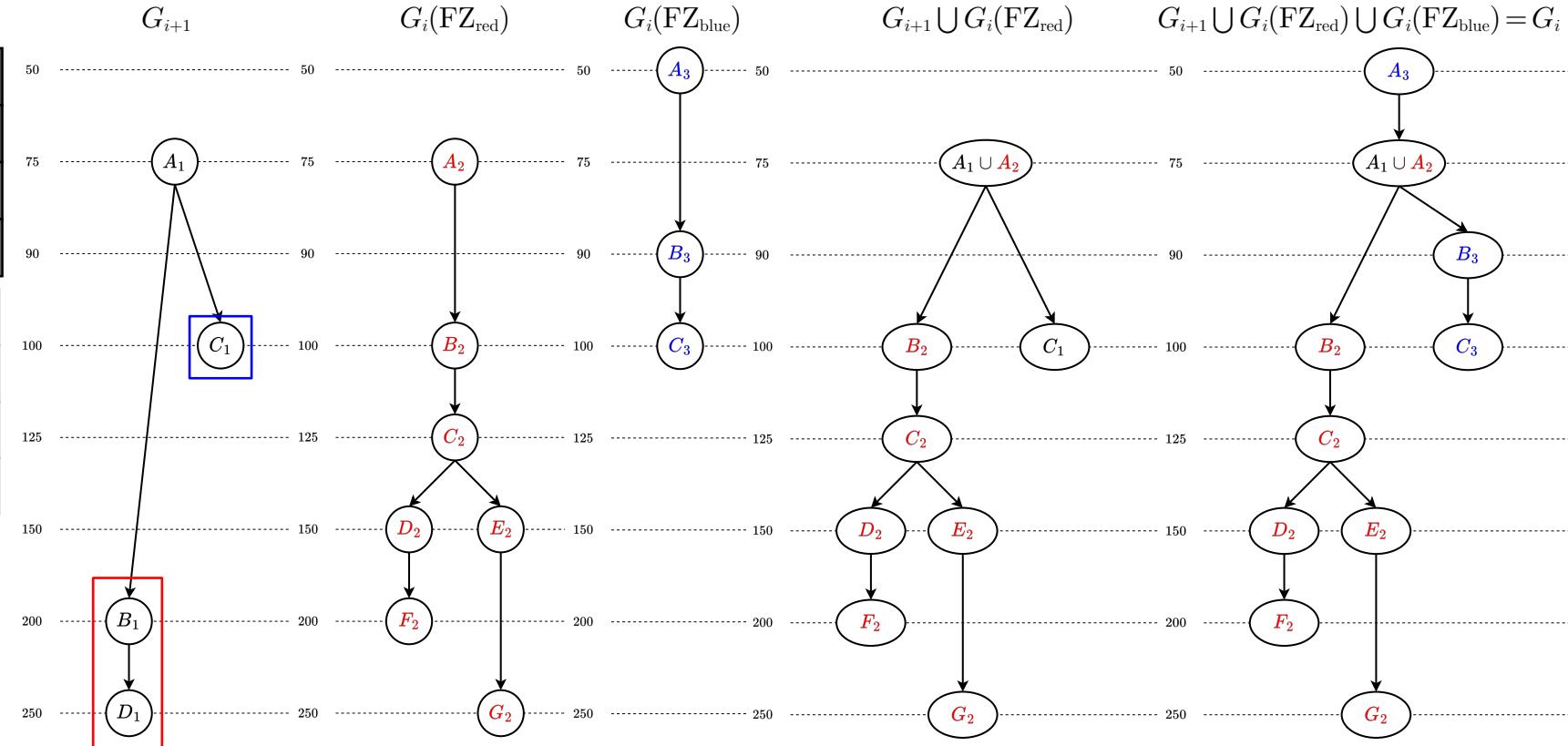


Appendix: merging hierarchies (example 1)

- Merging may introduce new gray levels lost during the downsampling process

f_i
75 100 150 150 75 75
200 125 250 150 75 75
200 125 75 75 50 100
150 125 75 75 90 100
75 100 150 150 75 75
200 125 250 150 75 75
200 125 75 75 50 100
150 125 75 75 90 100

f_{i+1}		
200 250 75		
200	75	100
200 250 75		
200	75	100



Appendix: merging hierarchies (example 2)

- Merging may create new branches from parent branches of upsampled nodes

f_i	f_{i+1}
100 250 250 200 200 50	
100 200 200 75 75 100	
50 50 50 100 200 200	
50 50 50 100 200 200	
50 50 50 200 50 50	
50 50 50 50 50 50	
100 250 250 200 200 50	
100 200 200 75 75 100	
50 50 50 100 200 200	
50 50 50 100 200 200	
50 50 50 200 50 50	
50 50 50 50 50 50	

G_{i+1}	$G_i(FZ_{red})$	$G_{i+1} \cup G_i(FZ_{red}) = G_i$
A_1	A_2	$A_1 \cup A_2$
B_1	B_2	B_2
C_1	C_2	C_2
D_1	D_2	D_2
E_1	E_2	E_2
	F_2	F_2
	G_2	G_2

