# Multi-Scale Component-Tree: A Hierarchichal Representation of Sparse Objects

Romain Perrin, Aurélie Leborgne, Nicolas Passat,
Benoît Naegel, and Cédric Wemmert

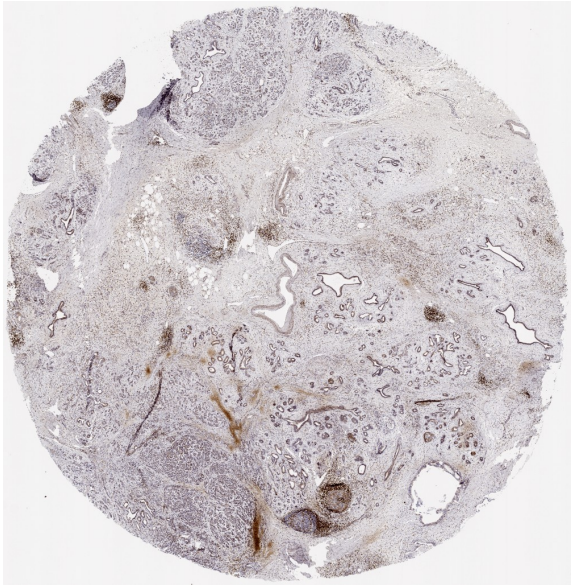
Firenze, 15-18 may 2024

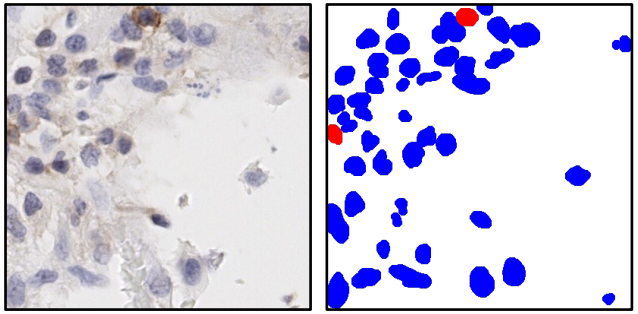
DGMM 2024

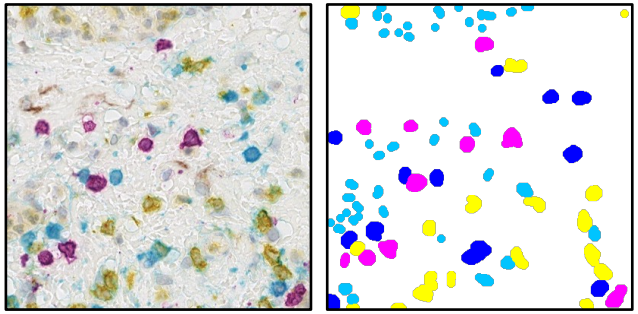
# Summary

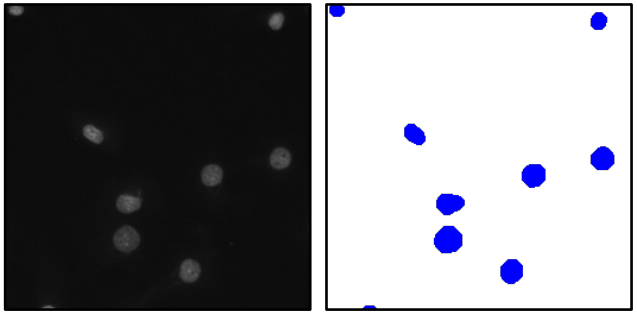
# I – Introduction: images

- 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







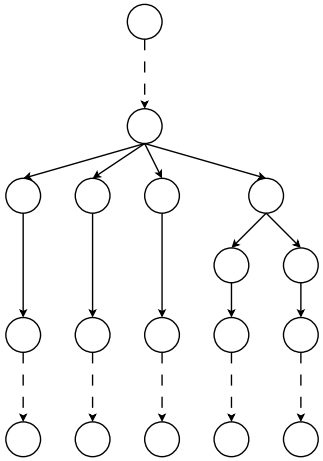


# I – Introduction: image representations

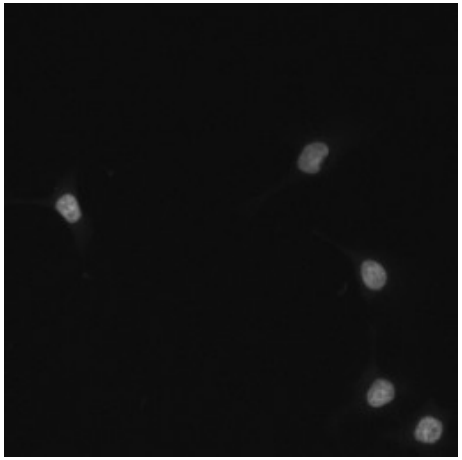
- **Pixel-based**
- **Block-based**
  - quadtree, octree...
- **Frequency domain**
  - Fourier transform, wavelets, ridgelets, contourlets...
- **Region-based**
  - superpixels, normalized cuts, Felzenszwalb & Huttenlocher, watershed...
- **Hierarchical**
  - Binary partition tree, component-tree, tree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

# I – Introduction: purpose

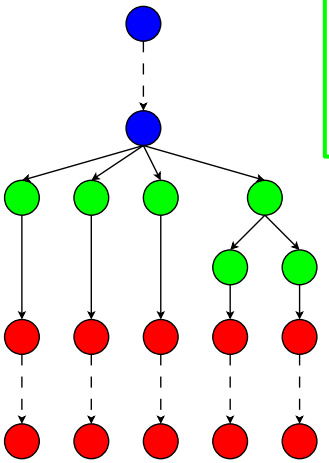
- Using the component-tree [1] as a basis
- Using different scales of the image
- Promote nodes with objects of interest to higher scales
- Keep other less relevant nodes at low scales

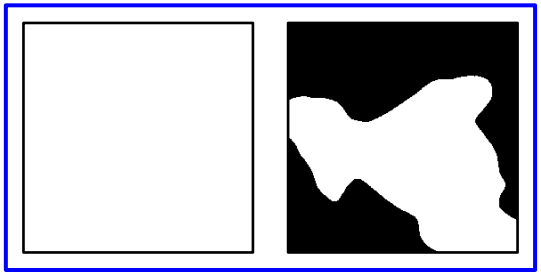


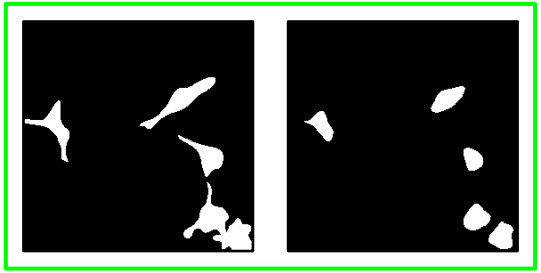
Single scale

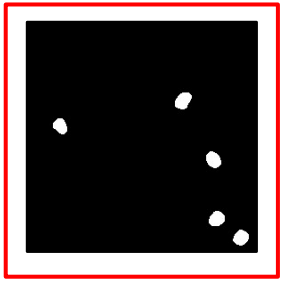


Multiple scales



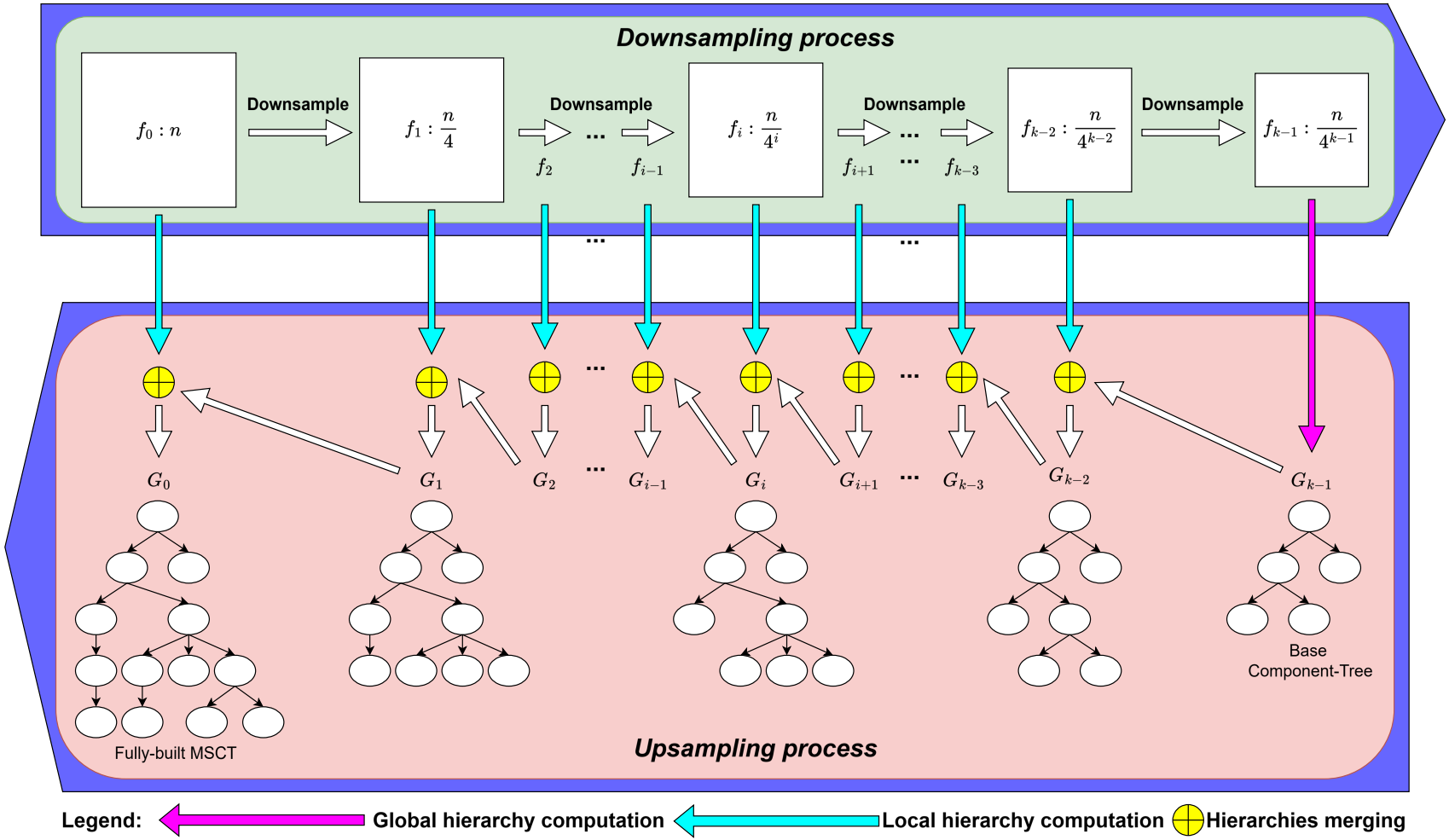






[1] P. Salembier, A. Oliveras, L. Garrido, Anti-extensive connected operators for image and sequence processing, IEEE Transactions on Image Processing, vol. 7, pp. 555–570, 1998.

# II – The MSCT: core principles



# II – The MSCT: construction algorithm

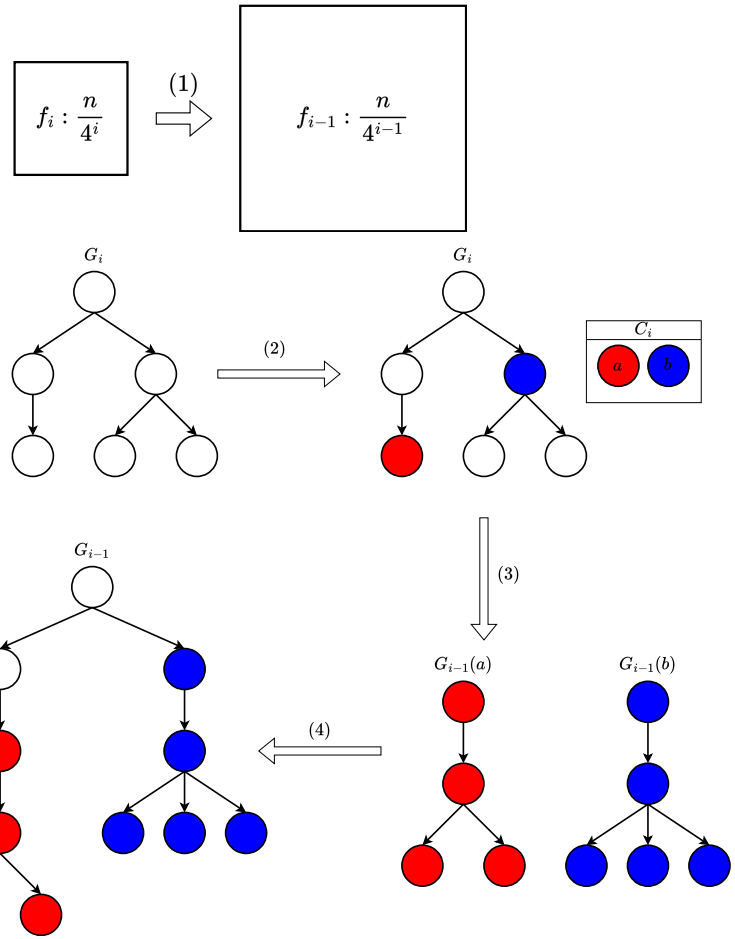
**Algorithm 1:** MSCT construction.

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

**begin**
  $F \longleftarrow \{f_0, \ldots, f_{k-1}\}$ with $f_0 = f$ and
   $\forall i \in [\![1, k-1]\!], f_i \longleftarrow \text{Downsample}(f_{i-1})$
  $G \longleftarrow G_{k-1} \longleftarrow \text{ComputeGlobalHierarchy}(f_{k-1})$
  **for** $i$ *from* $k-2$ *down to* $0$ **do**
    $C_i \longleftarrow \text{SelectNodes}(G, i)$
    **foreach** $N \in C_i$ **do**
      $G_i(N) \longleftarrow \text{ComputeLocalHierarchy}(f_i, N)$
      $\text{MergeHierarchies}(G, G_i(N), N)$
    **end**
  **end**
**end**

# II – The MSCT: operations

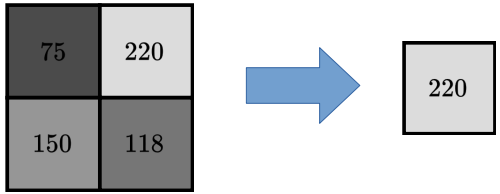
- Downsampling: $f_i \rightarrow f_{i+1}$
  - Downsampling operation (1)



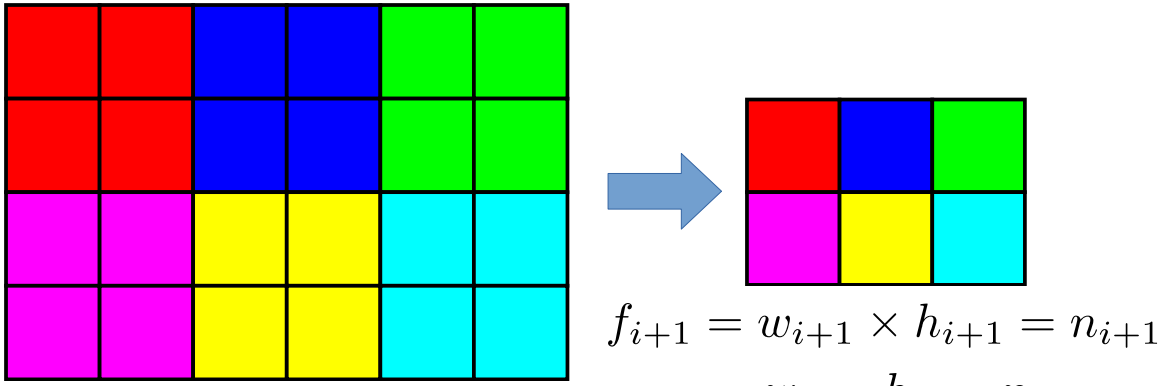

- Upsampling: $G_i \rightarrow G_{i-1}$
  - Computing a global hierarchy $G_{k-1}$
  - Selecting $C_i$ nodes from $G_i$ (2)
  - Computing local hierarchies of $C_i$ on $f_{i-1}$ (3)
  - Enriching $G_i$ to create $G_{i-1}$ (4)

# II – Building the MSCT: downsampling

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

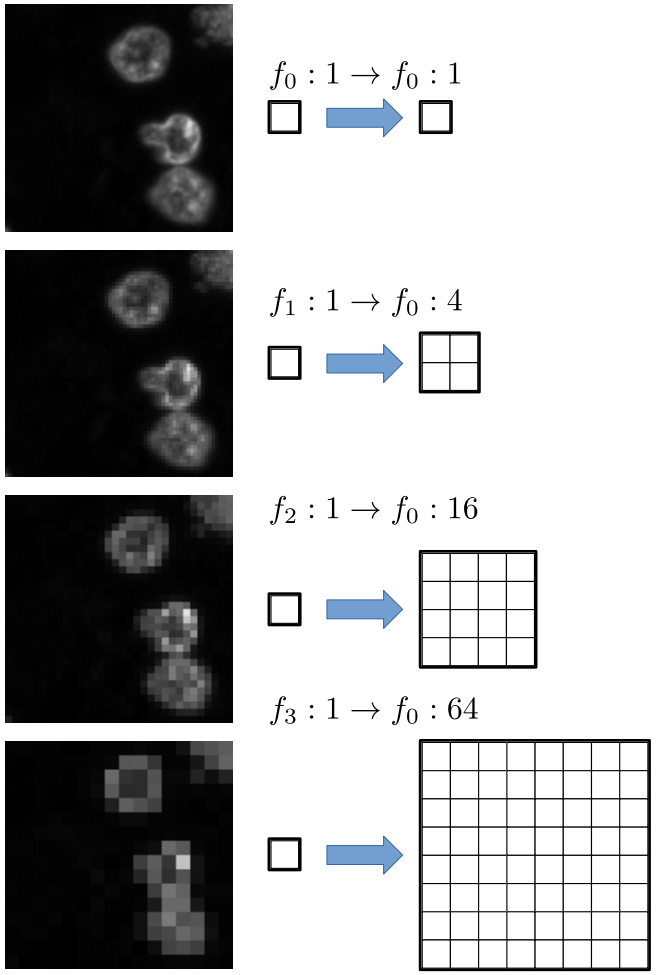



- ⇒ MaxPooling of size 2, stride 2

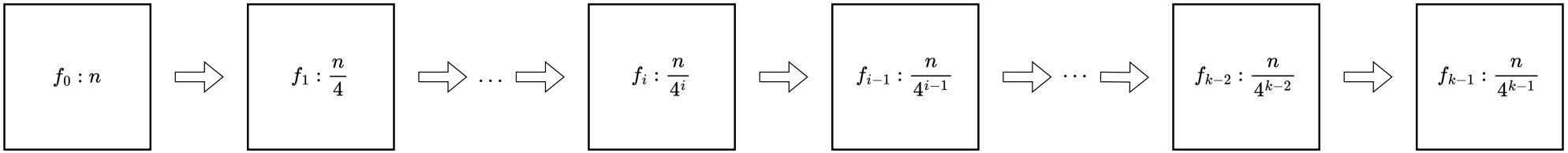


$$f_i = w_i \times h_i = n_i$$

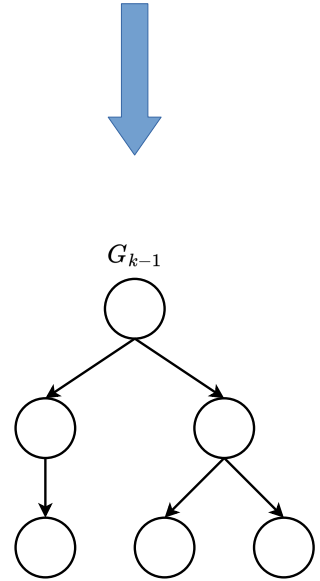
$$f_{i+1} = w_{i+1} \times h_{i+1} = n_{i+1} = \frac{w_i}{2} \times \frac{h_i}{2} = \frac{n_i}{4}$$



# 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 $G_{k-1} = (V_{k-1}, E_{k-1})$ : $\mathcal{O}(\frac{n}{4^{k-1}} \log(\frac{n}{4^{k-1}}))$
  - Najman & Couprie's algorithm [3]

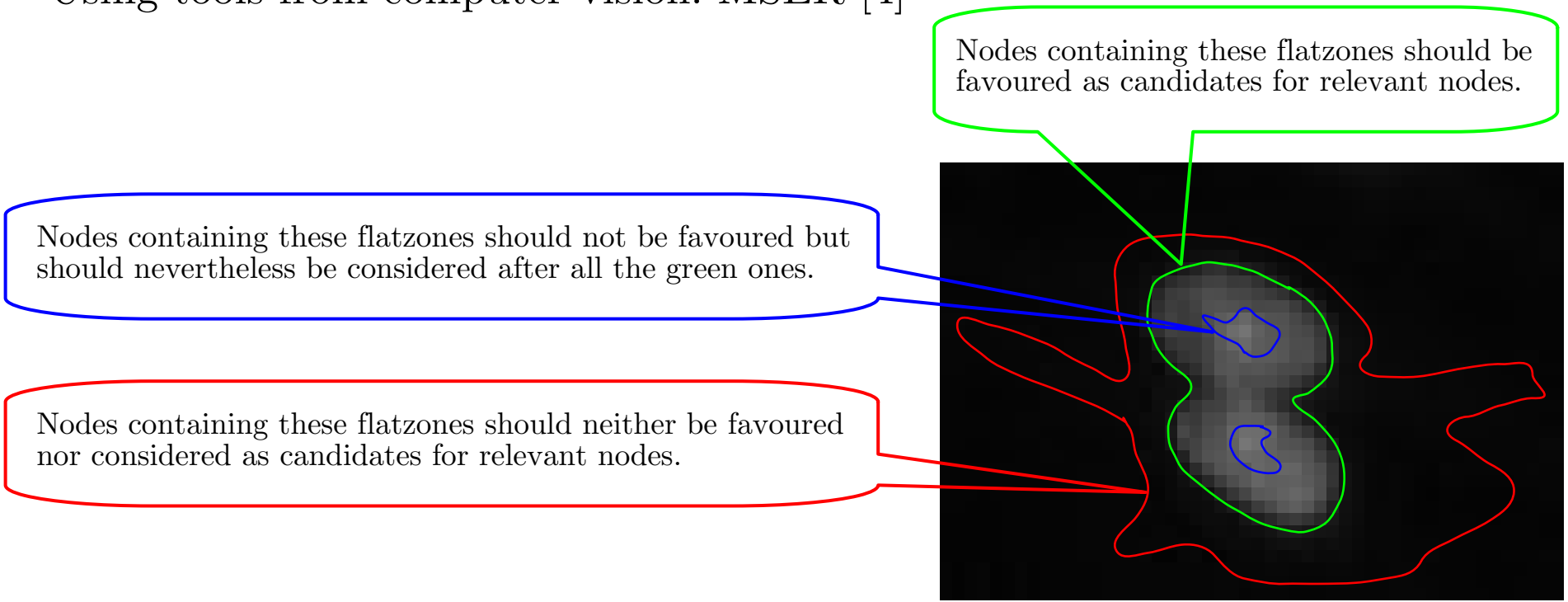




[2] E. Carlinet, T. Géraud, A comparative review of component tree computation algorithms, IEEE Transactions on Image Processing, vol. 23, pp. 3885-3895, 2014.
[3] L. Najman., M. Couprie, Building the component tree in quasi-linear time, IEEE Transactions on Image Processing, vol. 15, pp. 3531–3539, 2006.

# II – Building the MSCT: nodes metric

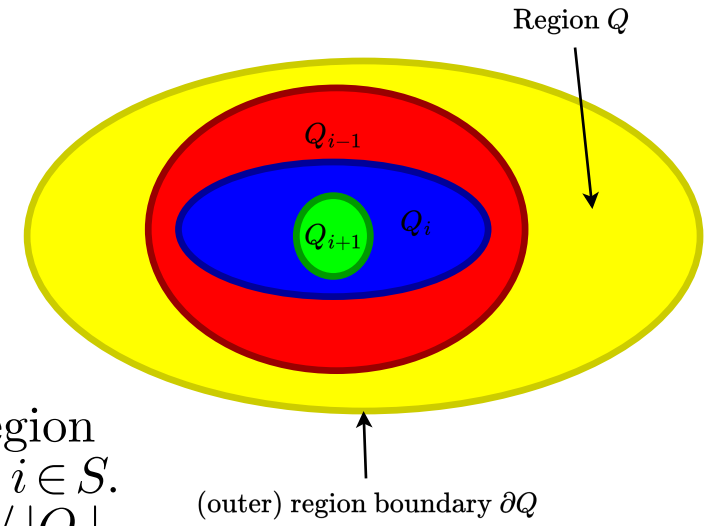
- A metric needs to be defined in order to select relevant nodes
- Using tools from computer vision: MSER [4]




[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.

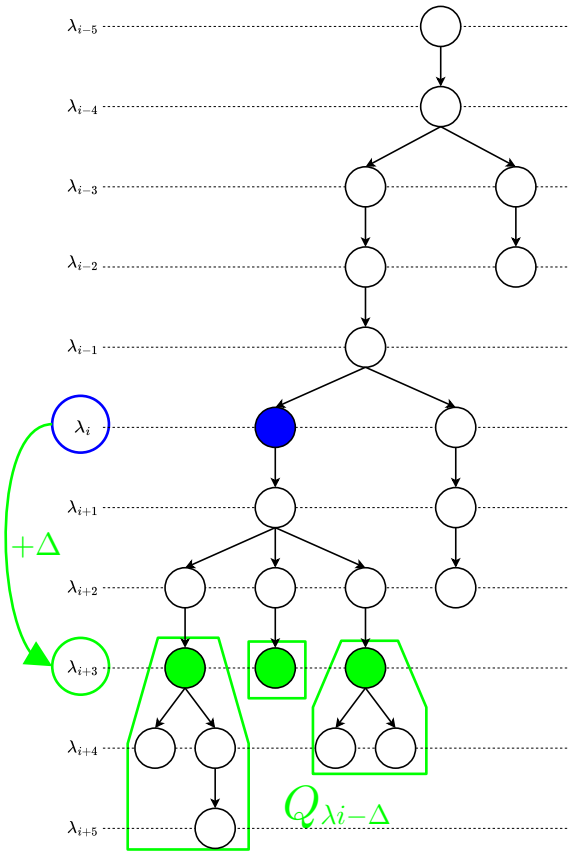
# II – Building the MSCT: MSER in theory

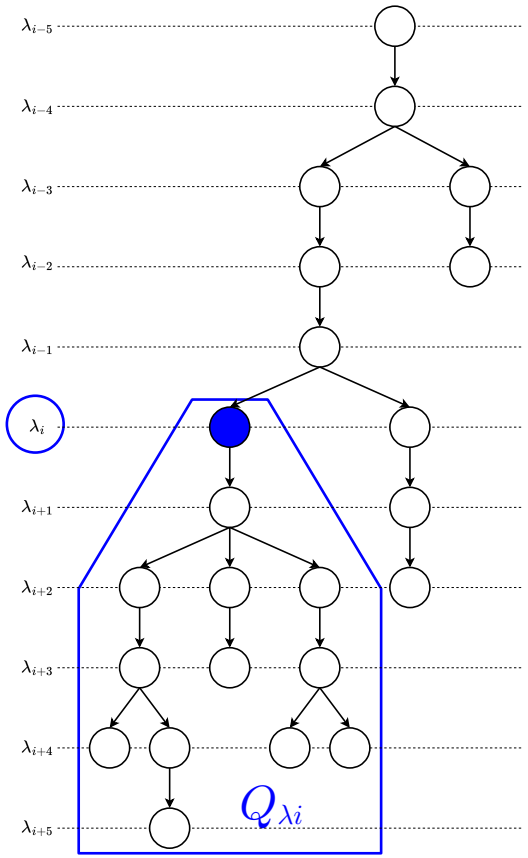
- **Image $I$** is a mapping $I : D \subset \mathbb{Z}^2 \rightarrow S$
  - S is totally ordered (total, antisymmetric and transitive relation $\leq$ exist)
  - An adjacency relation $A \subset D \times D$ exists denoting two adjacent points by $pAq$.
- **Region $Q$** is a connected subset of $D$.
  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 $i*$.

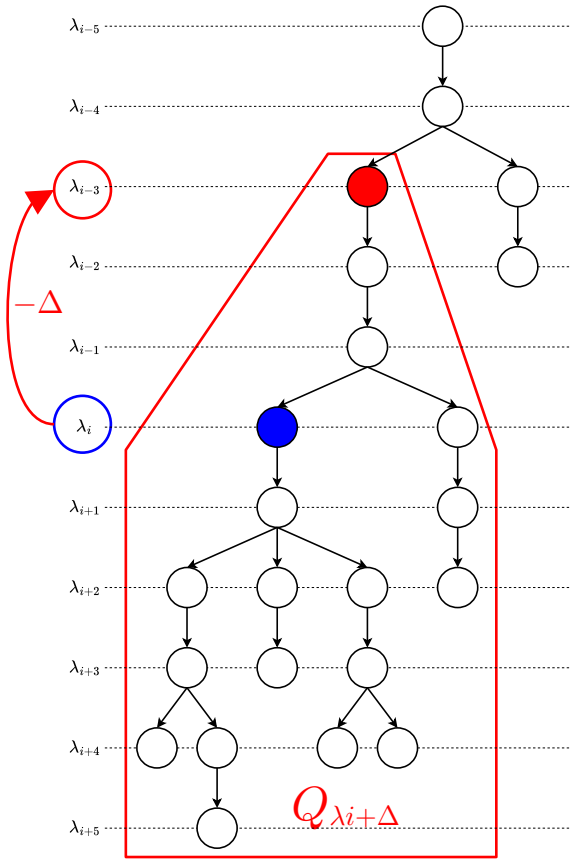


# II – Building the MSCT: MSER in practice

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



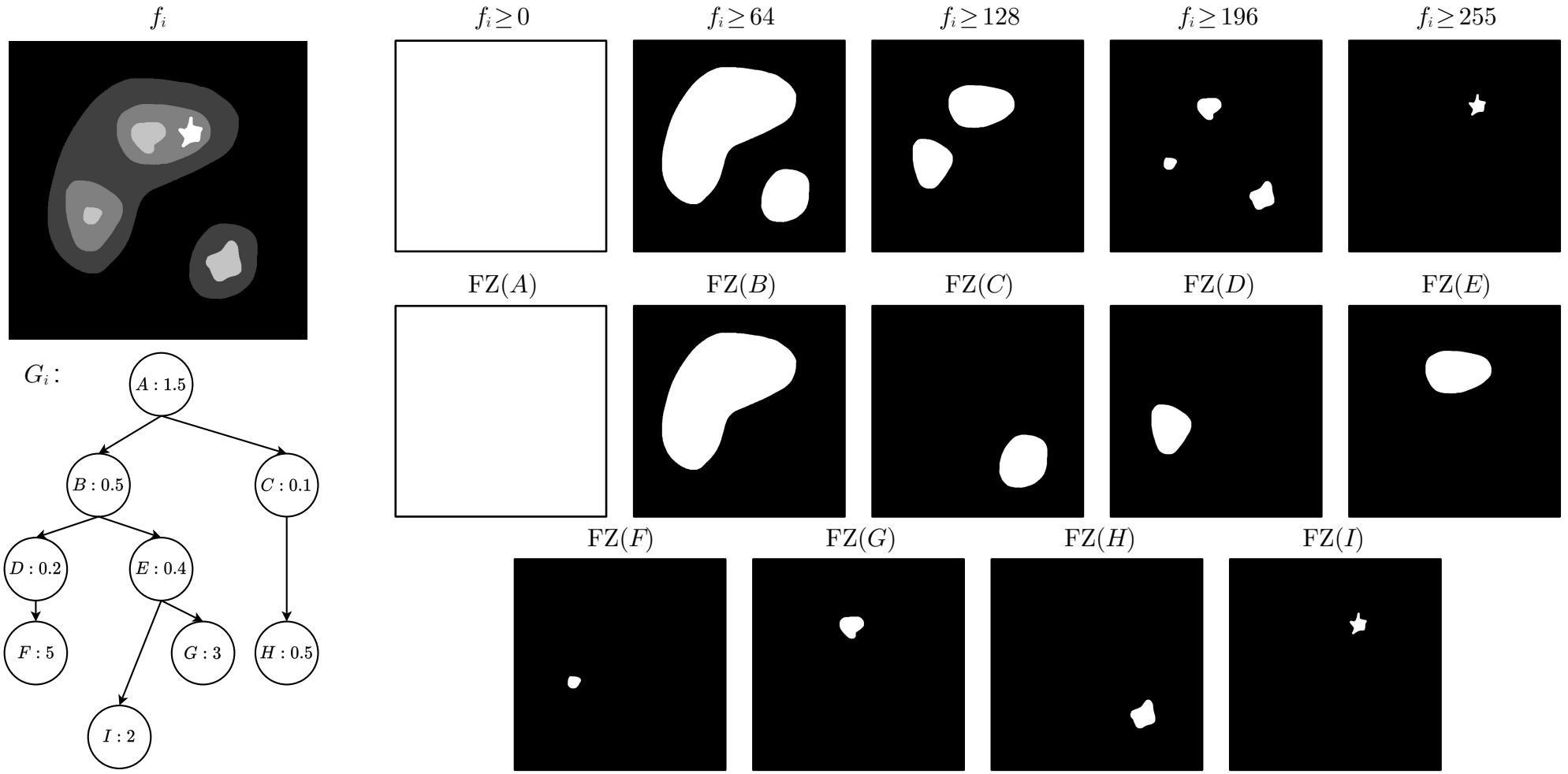
# II – Building the MSCT: selecting relevant nodes (algorithm)

**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 $G$

**begin**
  $C_i \longleftarrow \emptyset$
  $Q \longleftarrow \{v \in V_i \mid \forall a, b \in V_i, \mathrm{MSER}(a) \leq \mathrm{MSER}(b)\}$
  **while** $|Q| > 0$ **do**
    $v \longleftarrow \mathrm{PopFirst}(Q)$
    $C_i \longleftarrow C_i \cup \{v\}$
    $A_v \longleftarrow \mathrm{Ancestors}(G_i, v)$
    $D_v \longleftarrow \mathrm{Descendants}(G_i, v)$
    $Q \longleftarrow Q \setminus \{A_v \cup D_v\}$
  **end**
**end**

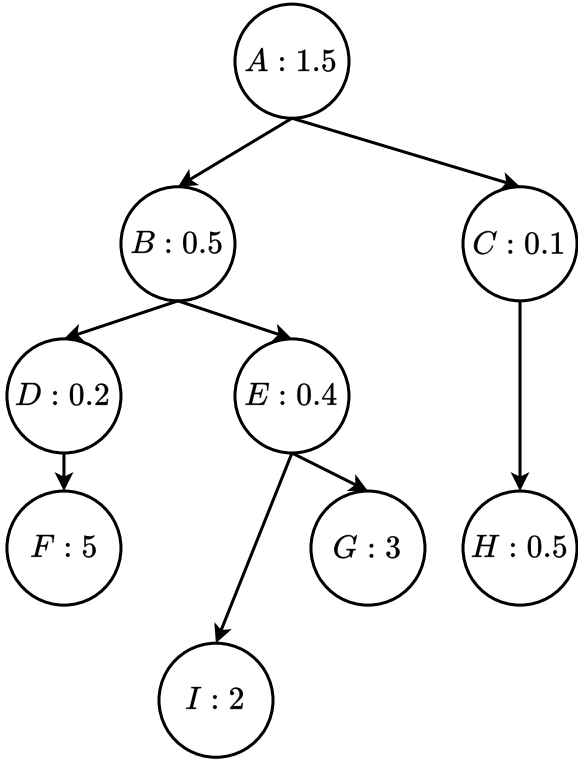
# II – Building the MSCT: selecting relevant nodes (example)



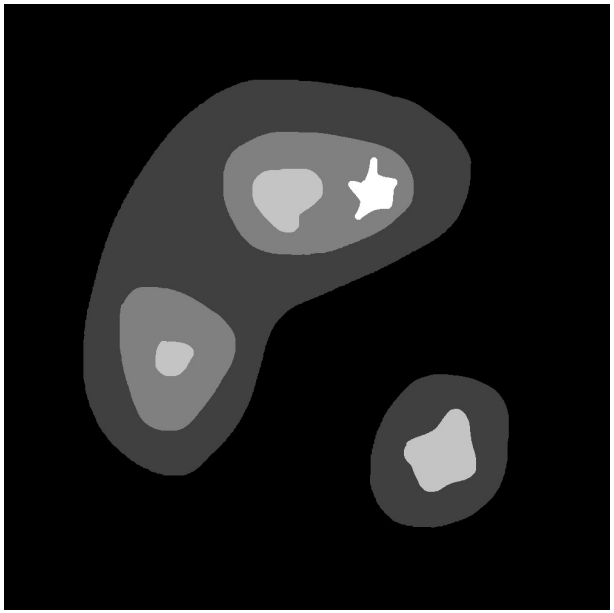
# II – Building the MSCT: selecting relevant nodes (example)

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

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




$G_i = (V_i, E_i)$



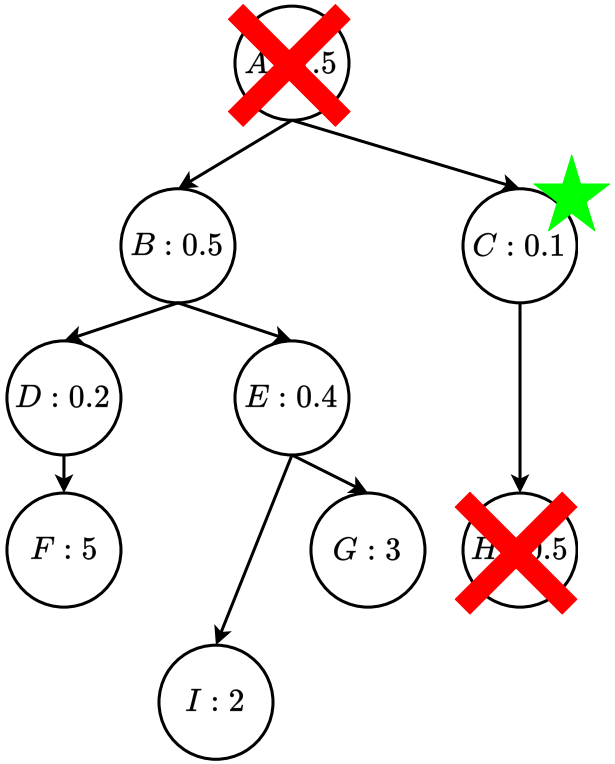
$f_i : w_i \times h_i = n_i$

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

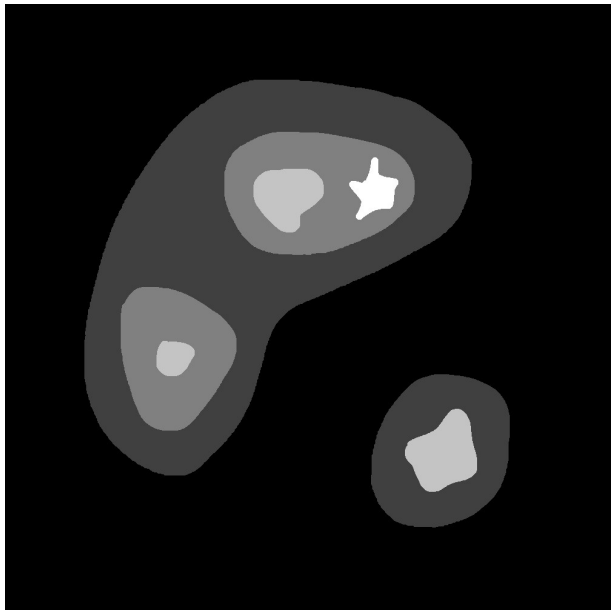
**Input:** $V_i = \{A, B, C, D, E, F, G, H, I\}$

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

**Iteration 1**
$Q = \{C, D, B, E, H, A, I, G, F\}$
$C_i = \{C\}$




$G_i = (V_i, E_i)$



$f_i : w_i \times h_i = n_i$

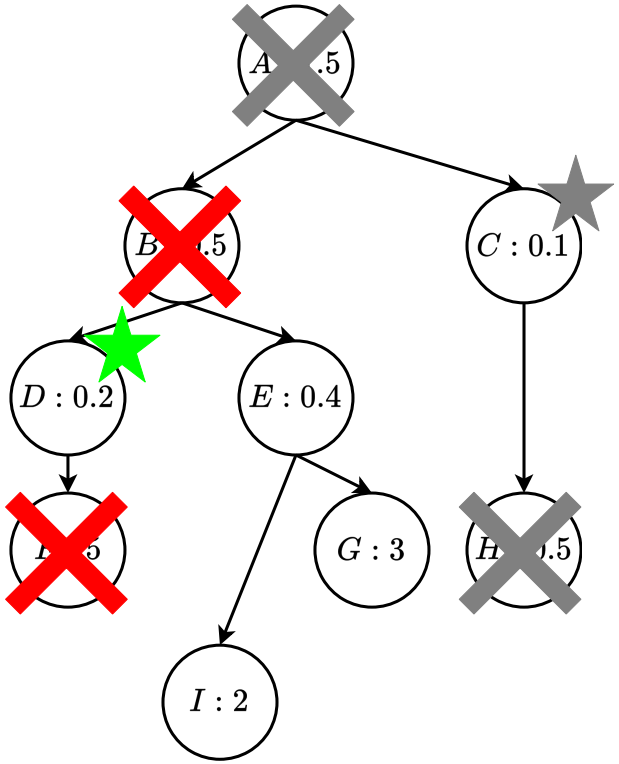
# II – Building the MSCT: selecting relevant nodes (example)

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

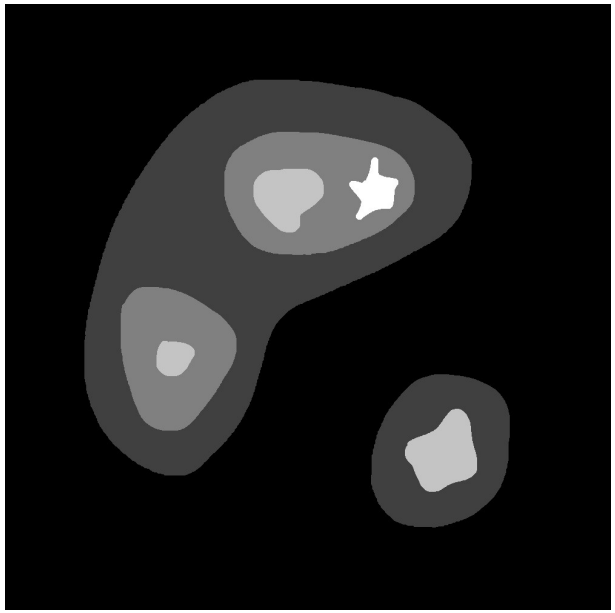
**Initialization**
$Q = \{C, D, B, E, H, A, I, G, F\}$
$C_i = \emptyset$

**Iteration 1**
$Q = \{C, D, B, E, H, A, I, G, F\}$
$C_i = \{C\}$

**Iteration 2**
$Q = \{D, B, E, I, G, F\}$
$C_i = \{C, D\}$




$G_i = (V_i, E_i)$



$f_i : w_i \times h_i = n_i$

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

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

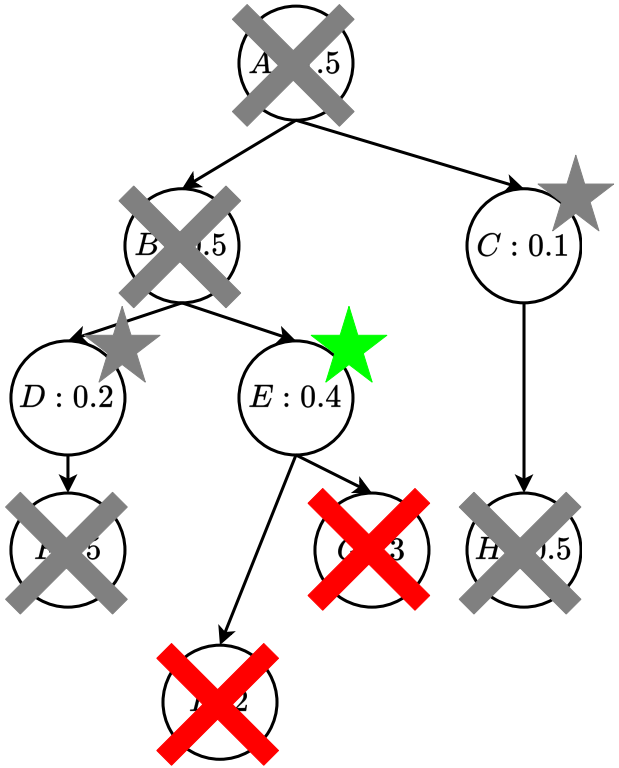
**Initialization**
$Q = \{C, D, B, E, H, A, I, G, F\}$
$C_i = \emptyset$

**Iteration 1**
$Q = \{C, D, B, E, H, A, I, G, F\}$
$C_i = \{C\}$

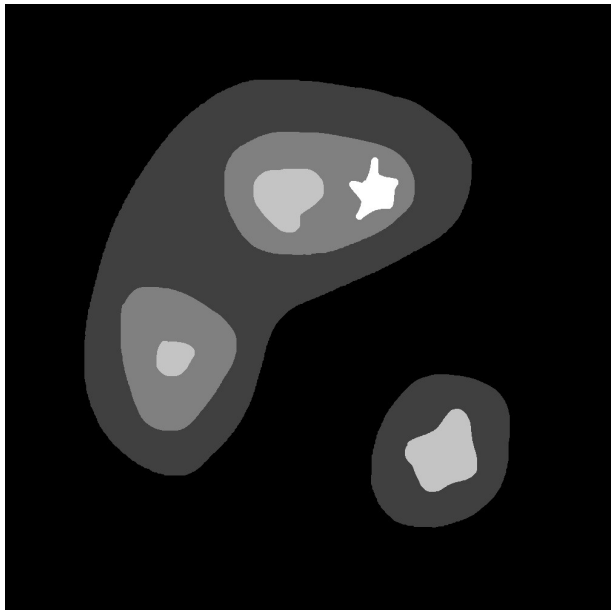
**Iteration 2**
$Q = \{D, B, E, I, G, F\}$
$C_i = \{C, D\}$

**Iteration 3**
$Q = \{E, I, G\}$
$C_i = \{C, D, E\}$

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




$G_i = (V_i, E_i)$



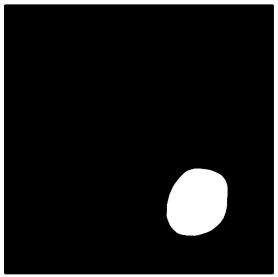
$f_i : w_i \times h_i = n_i$

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

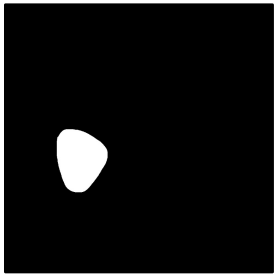
**Input:** $V_i = \{A, B, C, D, E, F, G, H, I\}$

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

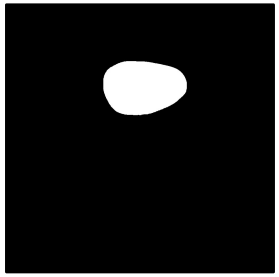
$\mathrm{FZ}(C)$

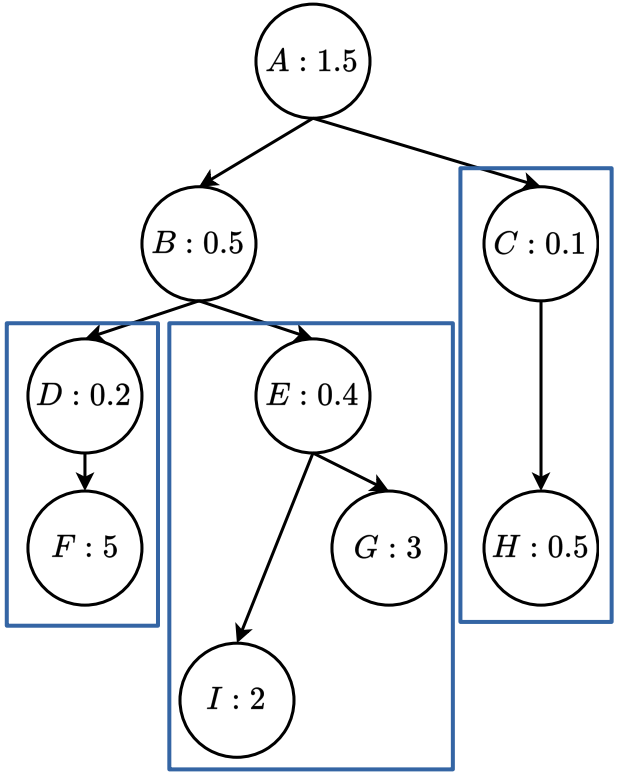


$\mathrm{FZ}(D)$

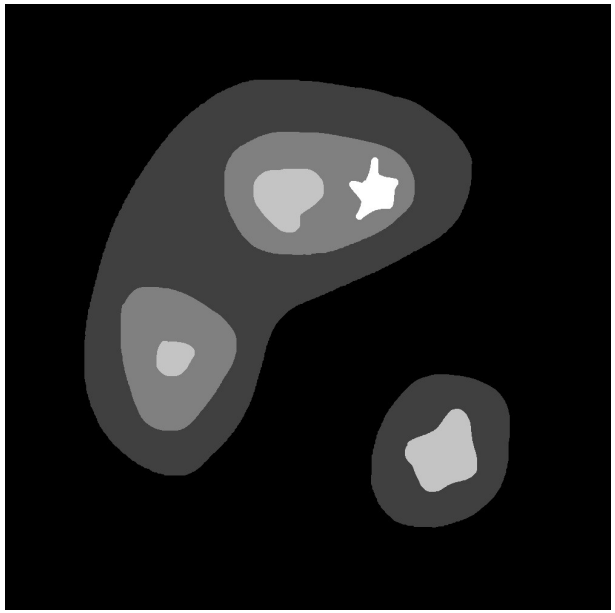


$\mathrm{FZ}(E)$





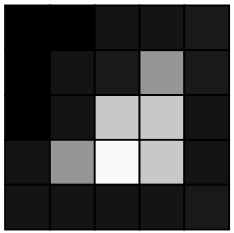

$G_i = (V_i, E_i)$



$f_i : w_i \times h_i = n_i$

# II – Building the MSCT: computing local hierarchies

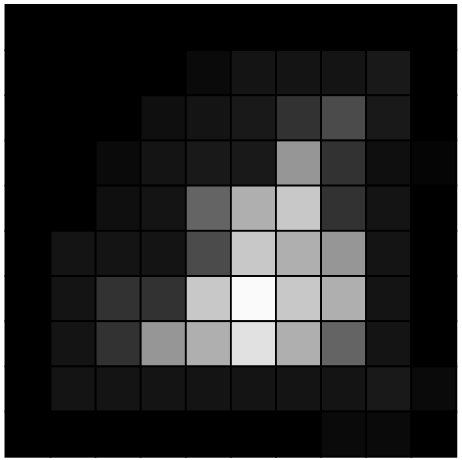
$f_i$

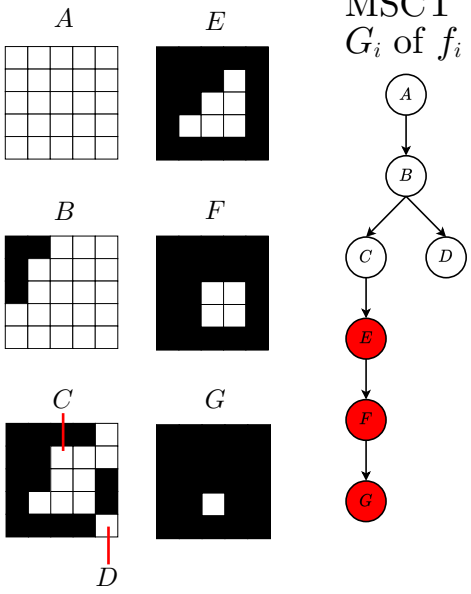


| | | | | |
|---|---|---|---|---|
| 0 | 0 | 20 | 20 | 25 |
| 0 | 20 | 25 | 150 | 25 |
| 0 | 20 | 200 | 200 | 20 |
| 20 | 150 | 250 | 200 | 20 |
| 20 | 20 | 20 | 20 | 25 |

$f_{i-1}$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 10 | 20 | 20 | 20 | 25 | 0 |
| 0 | 0 | 0 | 15 | 20 | 25 | 50 | 75 | 25 | 0 |
| 0 | 0 | 10 | 20 | 25 | 25 | 150 | 50 | 15 | 5 |
| 0 | 0 | 15 | 20 | 100 | 175 | 200 | 50 | 20 | 0 |
| 0 | 20 | 20 | 20 | 75 | 200 | 175 | 150 | 20 | 0 |
| 0 | 20 | 50 | 50 | 200 | 250 | 200 | 175 | 20 | 0 |
| 0 | 20 | 50 | 150 | 175 | 225 | 175 | 100 | 20 | 0 |
| 0 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 25 | 10 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 0 |

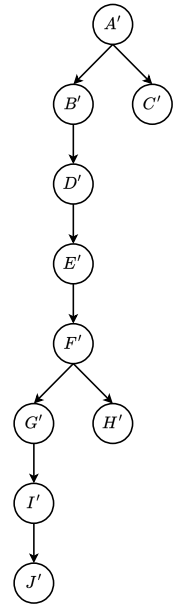


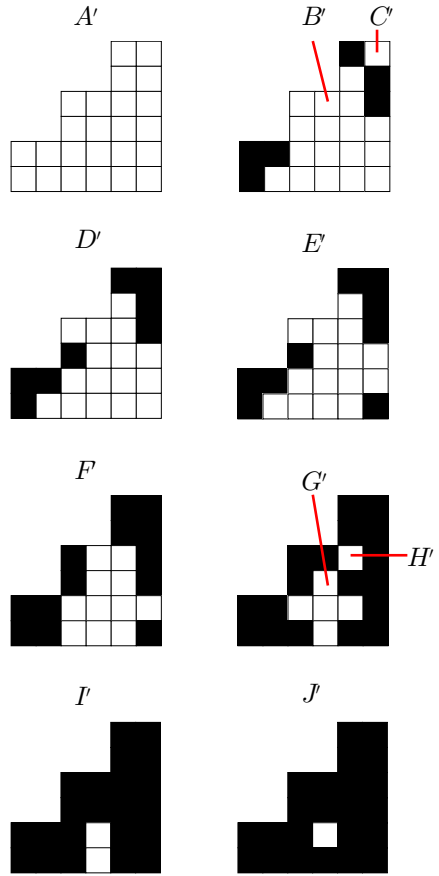



MSCT $G_i$ of $f_i$



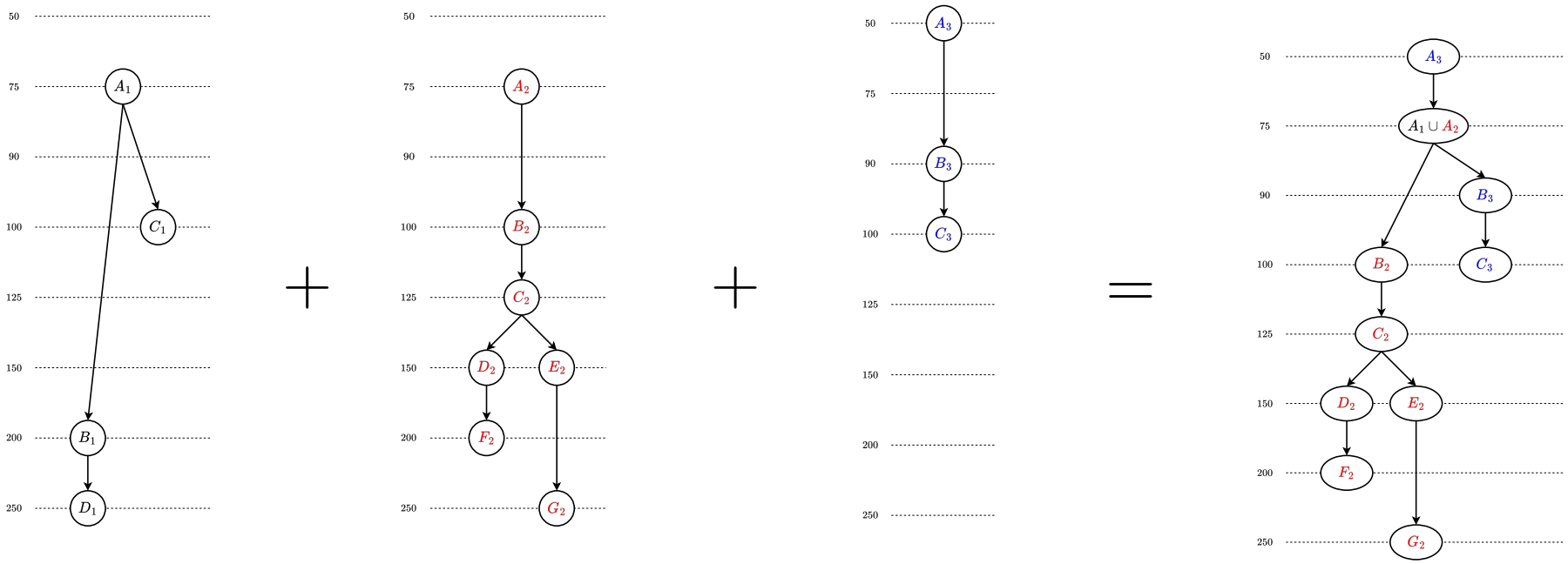

Partial component-tree $G_{i-1}(E)$ on $f_{i-1} \cap \mathrm{FZ}(E)$



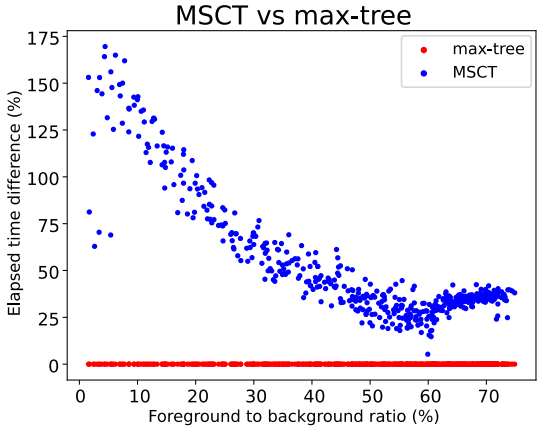
# II – Building the MSCT: merging hierarchies

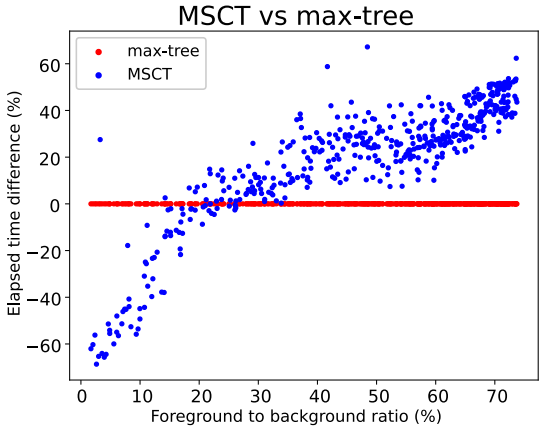
- Merging may introduce new nodes/gray-levels
- Flat zones may now represent unions of flat zones
- No redundancy concerning pixels

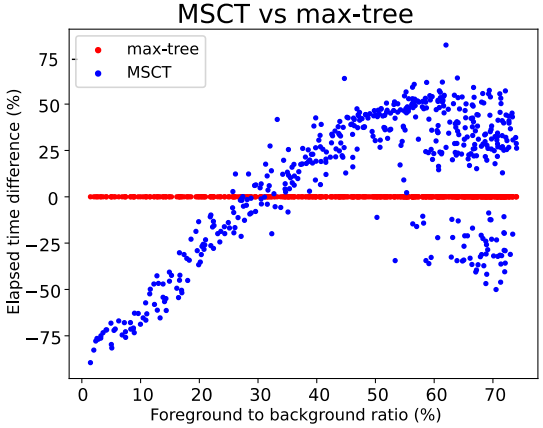


# II – Building the MSCT: Performances

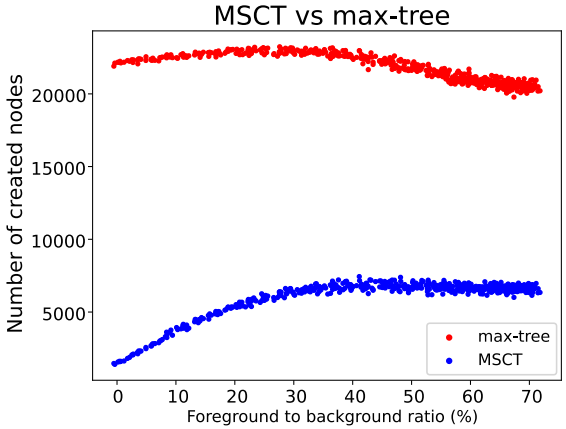
- Time computation measurements, $k = \{2,3,4\}$

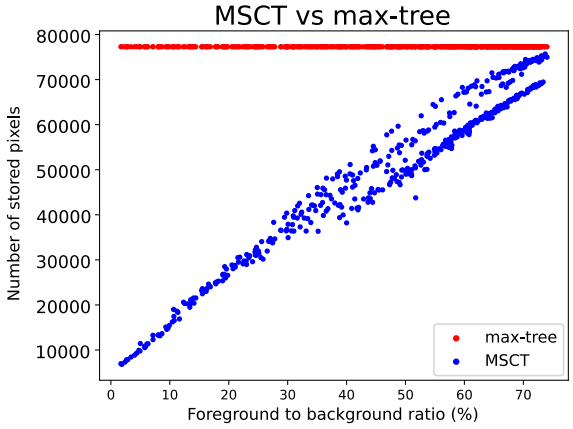



- Memory usage measurements, $k = 3$



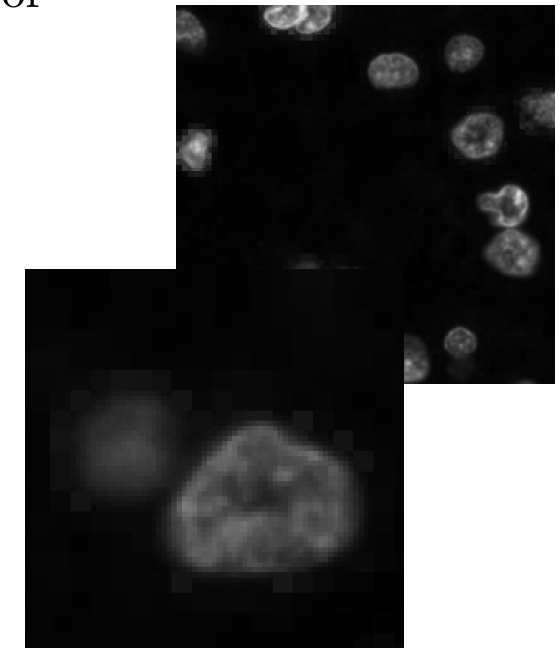
# III – Using the MSCT: segmentation (algorithm)

**Algorithm 1:** MSCT segmentation

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

**Result:** $P = \{P_0, \ldots, P_k\}$

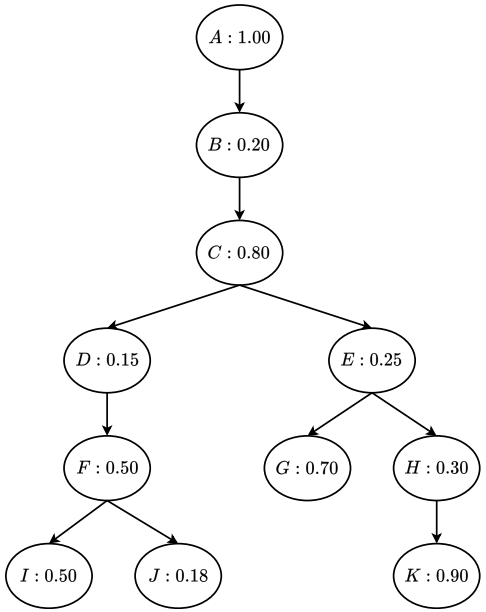
**begin**
  $P \longleftarrow \emptyset$
  $C_0 \longleftarrow \text{SelectNodes}(G)$
  $C'_0 \longleftarrow \text{FilterTree}(G, C_0, MSER_{max})$
  **for** $c \in C'_0$ **do**
    $c' \longleftarrow \text{FillHoles}(\text{Otsu}(\text{GaussianFilter}(c)))$
    $S \longleftarrow \text{Watershed}(f, c, \text{UltimateErosion}(c'))$
    $P \longleftarrow P \cup S$
  **end**
**end**



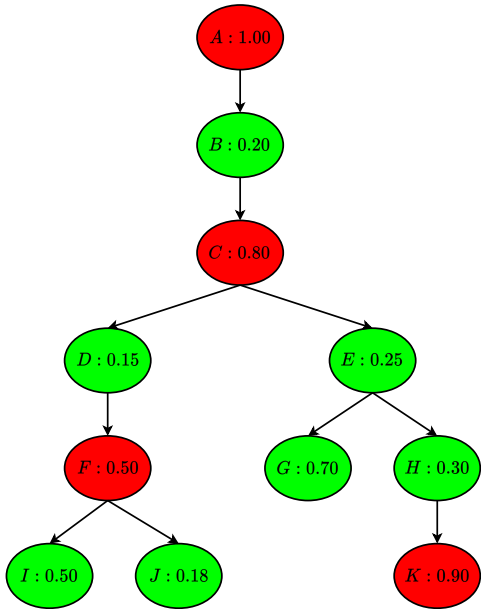
# III – Using the MSCT: segmentation (example)

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

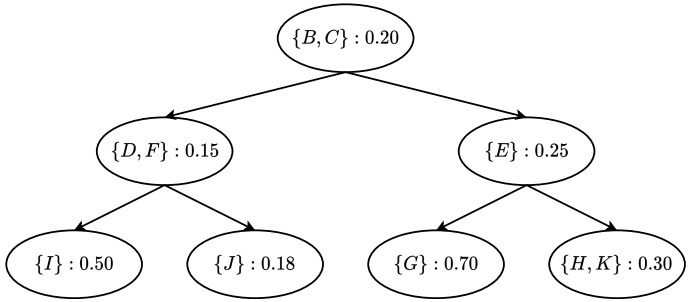
MSCT with MSER values




MSCT with MSER local branch minima



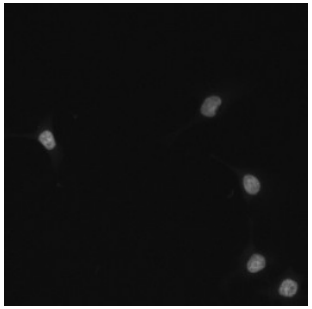

Simplified MSCT with each branch collapsed to its local MSER minima



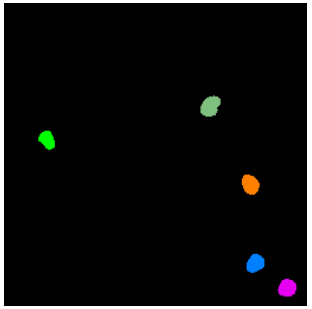
# III – Using the MSCT: segmentation (results)

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

Image 1



Ground truth 1



MSCT segmentation 1

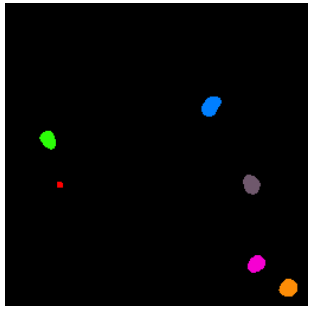
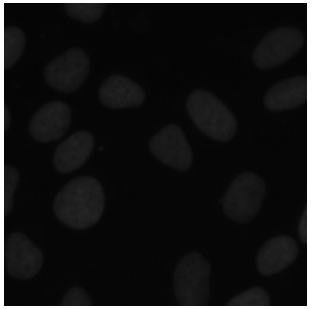


Image 3



Ground truth 3

MSCT segmentation 4

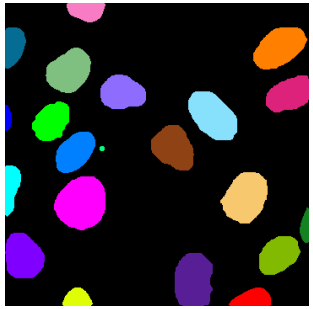
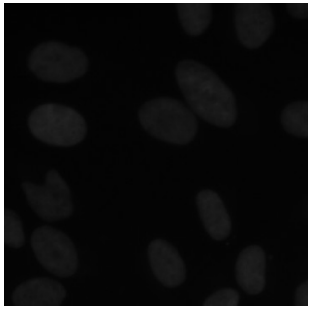
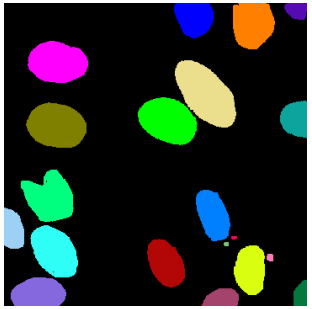


Image 2



Ground truth 2



MSCT segmentation 2

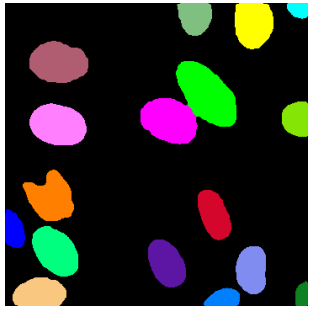
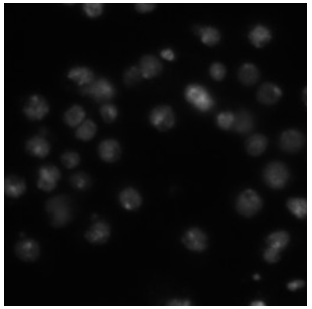
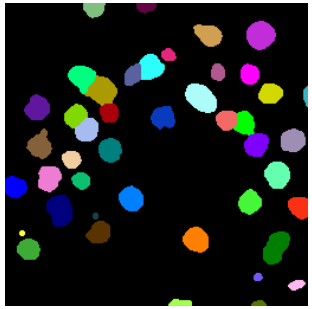


Image 4



Ground truth 4

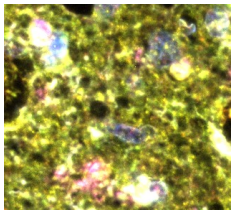


MSCT segmentation 4

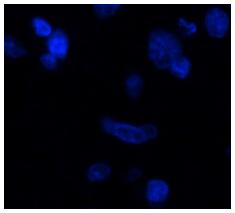
[5] https://www.kaggle.com/c/data-science-bowl-2018/

# III – Using the MSCT: segmentation & classification

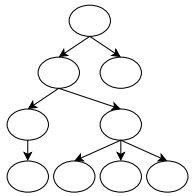
Multiplex image



DAPI channel

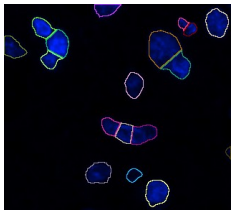


MSCT

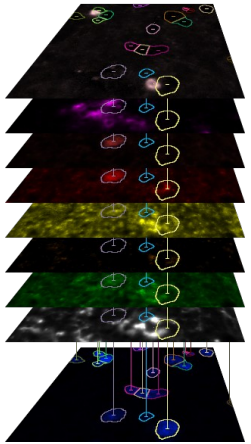


Segmentation

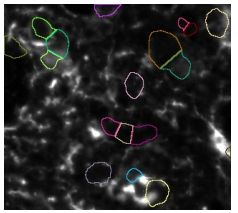
DAPI contours



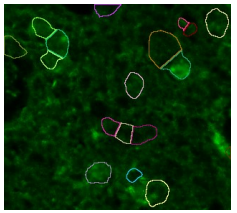
Projection



GFAP channel



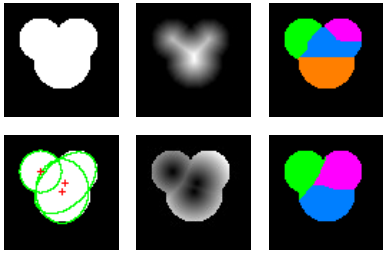
MS4A4A channel



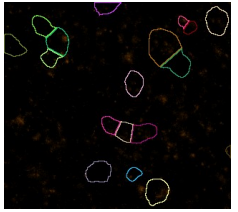
Feature vector : $c_i = \left[\sum_p f^j(p) \mid p \in n_i\right]_{j=1}^{k}$

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

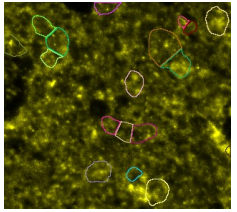
Watershed from EDT vs watershed from fake EDT with DTECMA [6]



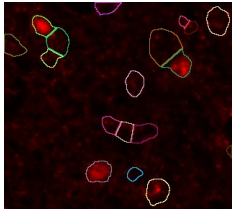
CD3 channel



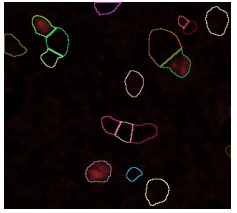
IDH1 channel



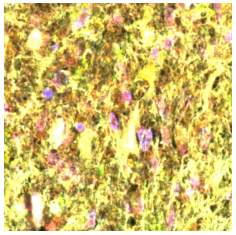
CD206 channel



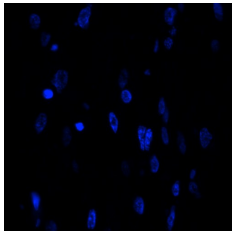
ATRX channel



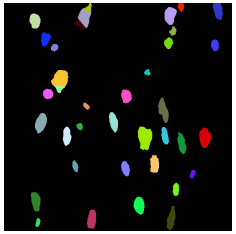
Multiplex 1



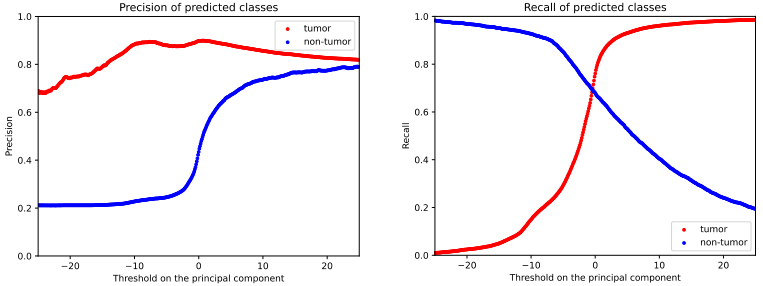
DAPI channel 1



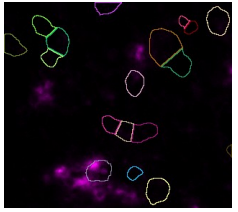
Segmentation 1



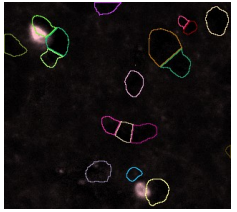
Precision and recall of binary classification (tumor/non tumor)



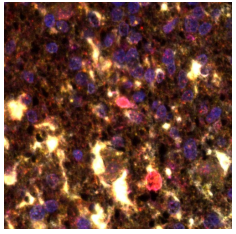

CD68 channel



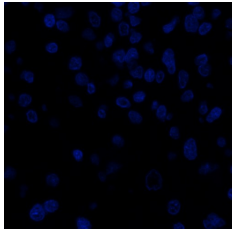
CD34 channel



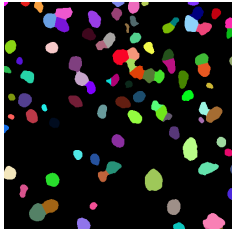
Multiplex 2



DAPI channel 2



Segmentation 2



[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.

# IV – Conclusion

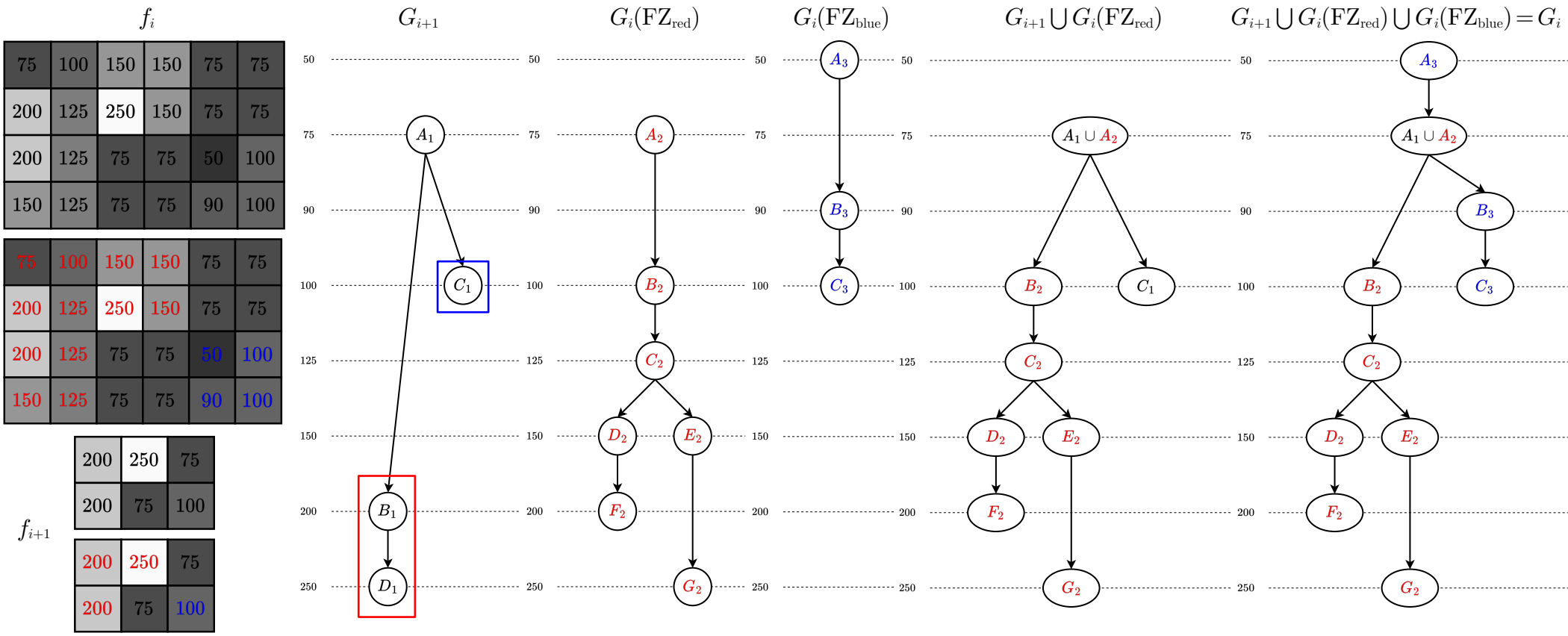
- Python code is available [7]
- Extension to dark on bright images (min-tree)
- Extension to colour images (MSCG for component-graphs [8])
- Generalization to other hierarchical structure
  - MSToS for the Tree of Shapes [9]
  - MS$\alpha$-T for the $\alpha$-tree or MS($\omega$)-T for the ($\omega$)-tree [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



# Appendix: merging hierarchies (example 2)

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