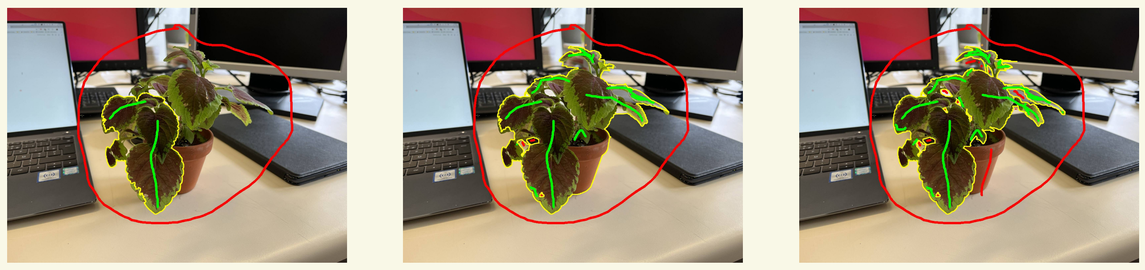
Incremental Watershed Cuts: Interactive Segmentation Algorithm with Parallel Strategy

Quentin Lebon[1], Josselin Lefèvre[1,2], Jean Cousty[1], Benjamin Perret[1]
[1] LIGM, Univ Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallée, France [2] Thermo Fisher Scientific, Bordeaux, France

Context

With large and complex images, segmentation can be a tedious task. For an easier and more pleasant experience for the user, interactive segmentation is often used.



The user is asked to place markers on the image to help the algorithm to know which object needs to be segmented

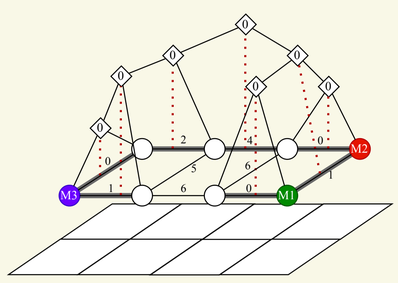
Problem

- Current state-of-the-art methods are not suitable for interactive segmentation.
  - They recompute the segmentation for the whole image at every interaction;
- Interactive segmentation of large images or volumes leads to huge latency between user clicks.
  - Example: MRI, astronomical, satellite images,

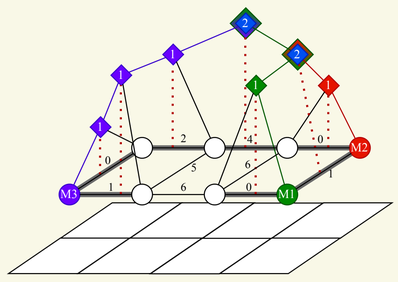
Solution: Incremental Watershed Cuts Algorithm

- At each interaction, leverage computations done at previous interactions;
- Produce the same result as the state-of-the-art algorithms but more responsive

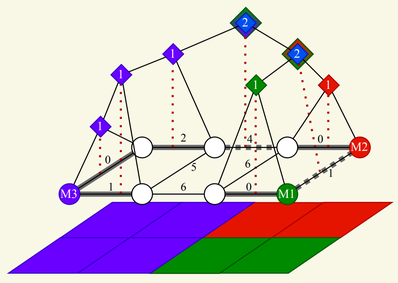
Incremental Watershed Cuts on Binary Partition Hierarchy



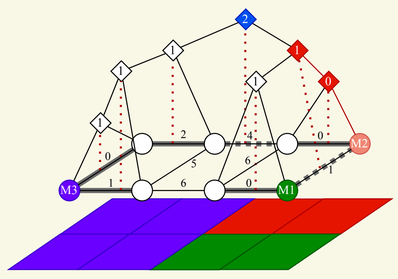
At first, markers are added on pixels of the image. In our hierarchy, leaf are marked as each vertex of the graph is linked to a pixel.



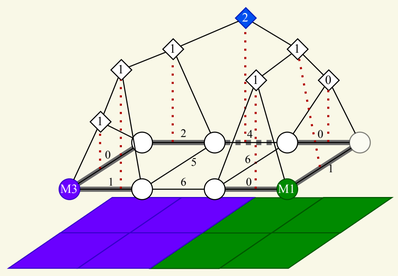
During markers addition (here $M1, 2, 3$), for each of them we climb the hierarchy bottom-up, each node encountered is incremented by 1. The process stopped is the root is reached, or a node is 2 after increment.



When a node equals 2, it is a watershed node. As each non-leaf node is linked to an edge, we can add those in the watershed cuts and remove them from the minimum spanning tree. Connected components are labeled accordingly.



During markers removal (here $M2$), for each of them we climb bottom-up, each node encountered is decremented by 1. The process stopped is the root is reached, or a node is 1 after decrement.



If a node equals 1, it is not a watershed node anymore. Each removed edge is added back in the minimum spanning tree, and the connected component are relabeled.

Sequential Strategy for Pixel Labeling

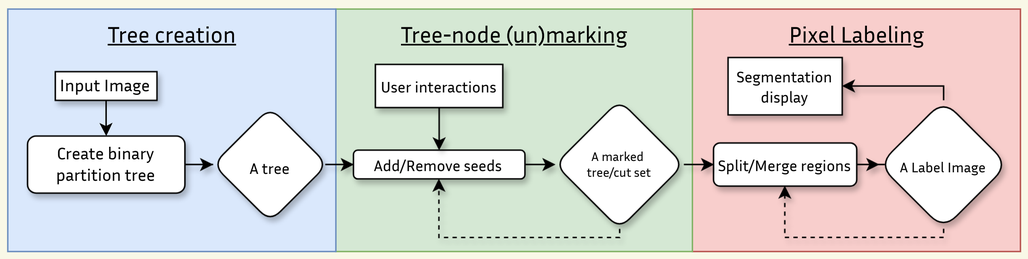
As we work with a minimum spanning tree :

- Removing an edge : split a connected component (CC) in two
- Adding an edge : merge two connected components into one

We label CC using Breadth First Search, by starting on the extremity of the added/removed edge. Thus, only the affected pixels are updated :

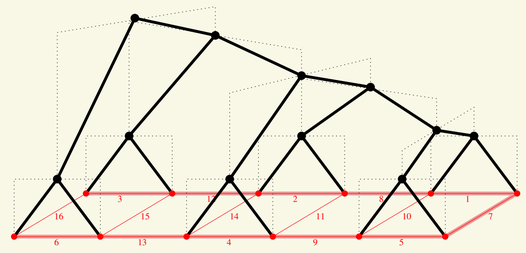
- When a merge occur, we can spread the label of the biggest CC to the other one
- However, this trick is not possible when we split CC

Overview of the Incremental Watershed Cuts Method



Binary Partition Hierarchy and Minimum Spanning Tree

The binary partition hierarchy is a data structure that can be obtained during the execution of Kruskal's algorithm, allowing to obtain a minimum spanning tree A of a given $G = (V, E, w)$. Each leaf is associated to a vertex of A and each internal node is an edge of A.



In black a binary partition tree and in bold red its associated minimum spanning tree.

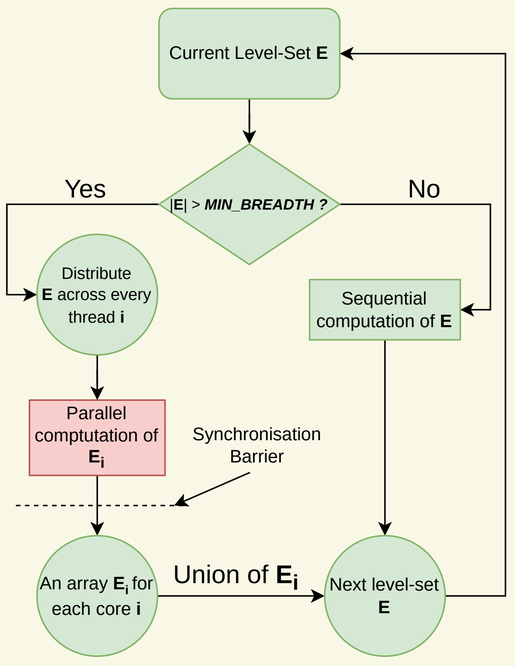
Problem - Efficient Labeling of Minimum Spanning Forest

- The average execution time at each user interaction is :
  - less than 0.10sec on 2D images
  - 0.50 to 2.50sec on 3D images
- On 3D data, our method is not responsive enough
- On large 3D images, 3D data CC labeling take up to 99% of the time for a single user interaction.

Solution: Parallel Pixel Labeling

- Took inspiration from [1] and adapt the principle to our problem

Parallel Strategy for Pixel Labeling



- At beginning, the level set E is composed of the extremity of the added/removed edge
- If $|E|$ is above a threshold `MIN_BREADTH` we explore in parallel :
  - E is equally distributed in small subset for each thread i, thus they have the same amount of work
  - Each thread label the successors of its level-set, and create the next level-set E_i at the same time
  - Once all thread are finish all E_i are merged into one array E, which is the global next level-set
- Exploration is done when the current level-set is empty.
- `MIN_BREADTH` allows us to explore small level-set in sequential, thus avoid bad performance because of synchronization time

Experiences and results

- We generate marker automatically to simulate user interaction. Our experiments are based on :
  - Natural 2D images dataset with 150 images from 3.2 to 18 MPixels
  - Liver segmentation 3D images dataset with 5 MRI of 32.5 to 68 MPixels
- We compared ourselves with state-of-the-art methods and implementation

Method	Init	Average	Min	Accumulated
IWS*	7.92	86.18	26.78	0.54
NIWS*	7.92	24.72	23.71	0.32
IWS_PAR*	7.34	216.78	96.14	0.89
Amira	∅	8.14	7.38	0.13
ITK	∅	6.49	6.22	0.11

Result on the 3D data-set (* ours)

- The scalability of the methods is confirmed, performance increase as the user interact
- On 3D data using the parallel method, a user interaction is proceeded in 0.15 to 0.7sec

References

[1] Youkana, Imane and Cousty, Jean and Saouli, Rachida and Akil, Mohamed. Parallelization strategy for elementary morphological operators on graphs: distance-based algorithms and implementation on multicore shared-memory architecture. JMIV, pages 136–160, 2017
[2] Lebon, Q., Lefevre, J., Cousty, J., and Perret, B. (2023). Interactive Segmentation with Incremental Watershed Cuts. CIRAP pages 189–200
[3] Lebon, Q., Lefevre, J., Cousty, J., and Perret, B. (2024). Incremental Watershed Cuts: Interactive Segmentation Algorithm with Parallel Strategy. Under Review