university of groningen

# A parallel implementation of the dual-input Max-Tree algorithm for attribute filtering

Georgios K. Ouzounis and Michael H.F. Wilkinson
*Institute of Mathematics and Computing Science*
*University of Groningen*
georgios@cs.rug.nl, m.h.f.wilkinson@rug.nl

ISMM 2007, 8-10 October, Rio de Janeiro

# Introduction

- Many image processing operators allow easy parallellization through either locality or separability
- Connected operators are neither separable nor local
- Recently, a parallellization strategy for Max-Tree-based attribute filtering was developed in Groningen
- It is based on a simple strategy:
  - Compute separate Max-Trees for $N_p$ slices of an image or volume
  - Merge the trees into a single tree
  - Filter the resulting tree using multiple CPUs concurrently
- Speed-up of up to 14 on 16 MIPS CPUs was obtained.
- In this presentation an extension to the Dual-Input Max-Tree algorithm for second-generation connectivity is proposed

# Attribute filters

- Binary attribute filters are based on trivial filters $\Gamma_T$ which are defined as

$$\Gamma_T(C) = \begin{cases} C & \text{if } T(C) \text{ is true} \\ \emptyset & \text{otherwise,} \end{cases} \tag{1}$$

in which $T$ is a criterion which usually takes the form $T(C) = (Attr(C) \geq \lambda)$, and $C \in \mathcal{C}$.

- The binary attribute opening $\Gamma^T$ of set $X$ with increasing criterion $T$ is given by

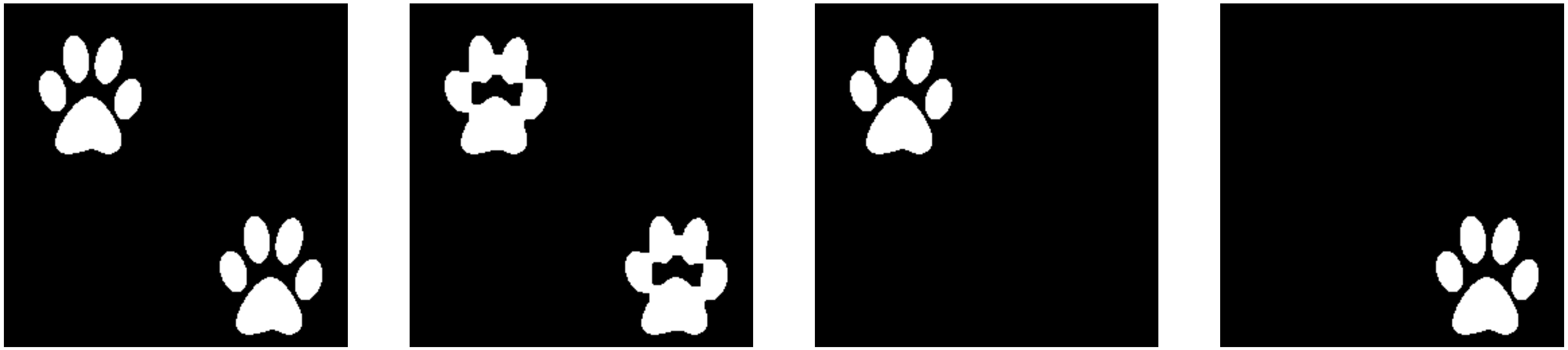
$$\Gamma^T(X) = \bigcup_{x \in X} \Gamma_T(\Gamma_x(X)) \tag{2}$$

# Second-Generation Connectivity

- Second-generation connectivities use an operator $\psi$ which modifies $X$ and a base class $\mathcal{C}$.
- The resulting connectivity class is denoted $\mathcal{C}^\psi$ and corresponds to a second-generation connected opening $\Gamma_x^\psi$.
- The connected opening $\Gamma_x^\psi$ for a second-generation connectivity based on $\psi$ of image $X$ is

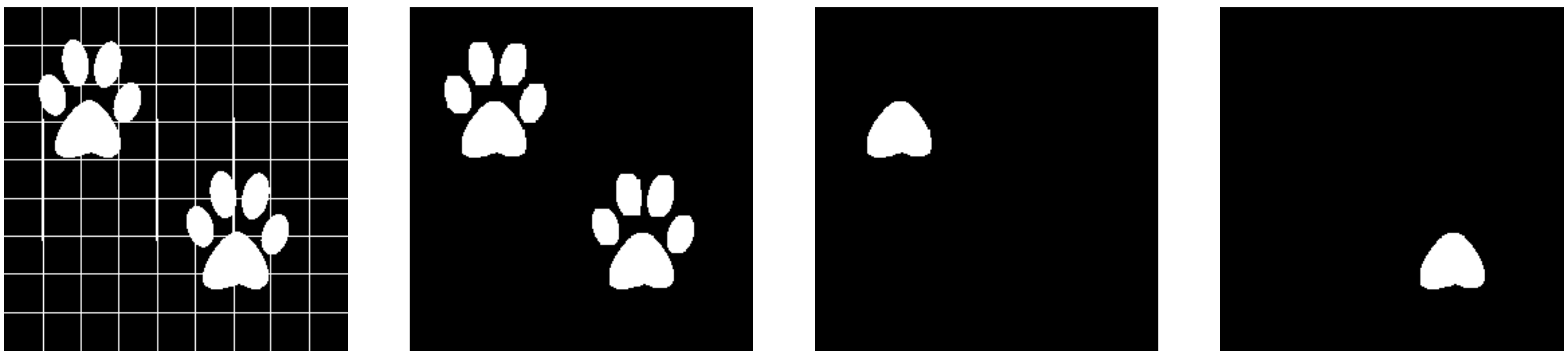
$$\Gamma_x^\psi(X) = \begin{cases} \Gamma_x(\psi(X)) \cap X & \text{if } x \in X \cap \psi(X) \\ \{x\} & \text{if } x \in X \setminus \psi(X) \\ \emptyset & \text{otherwise,} \end{cases} \tag{3}$$

  in which $\Gamma_x$ is the connected opening based on $\mathcal{C}$.
- If $\psi$ is extensive $\mathcal{C}^\psi$ is *clustering* based.
- If $\psi$ is anti-extensive $\mathcal{C}^\psi$ is *partitioning* based.



$X$ $\psi(X)$ $\Gamma_p^{\psi}(X)$ $\Gamma_q^{\psi}(X)$

**Clustering-based connectivity**

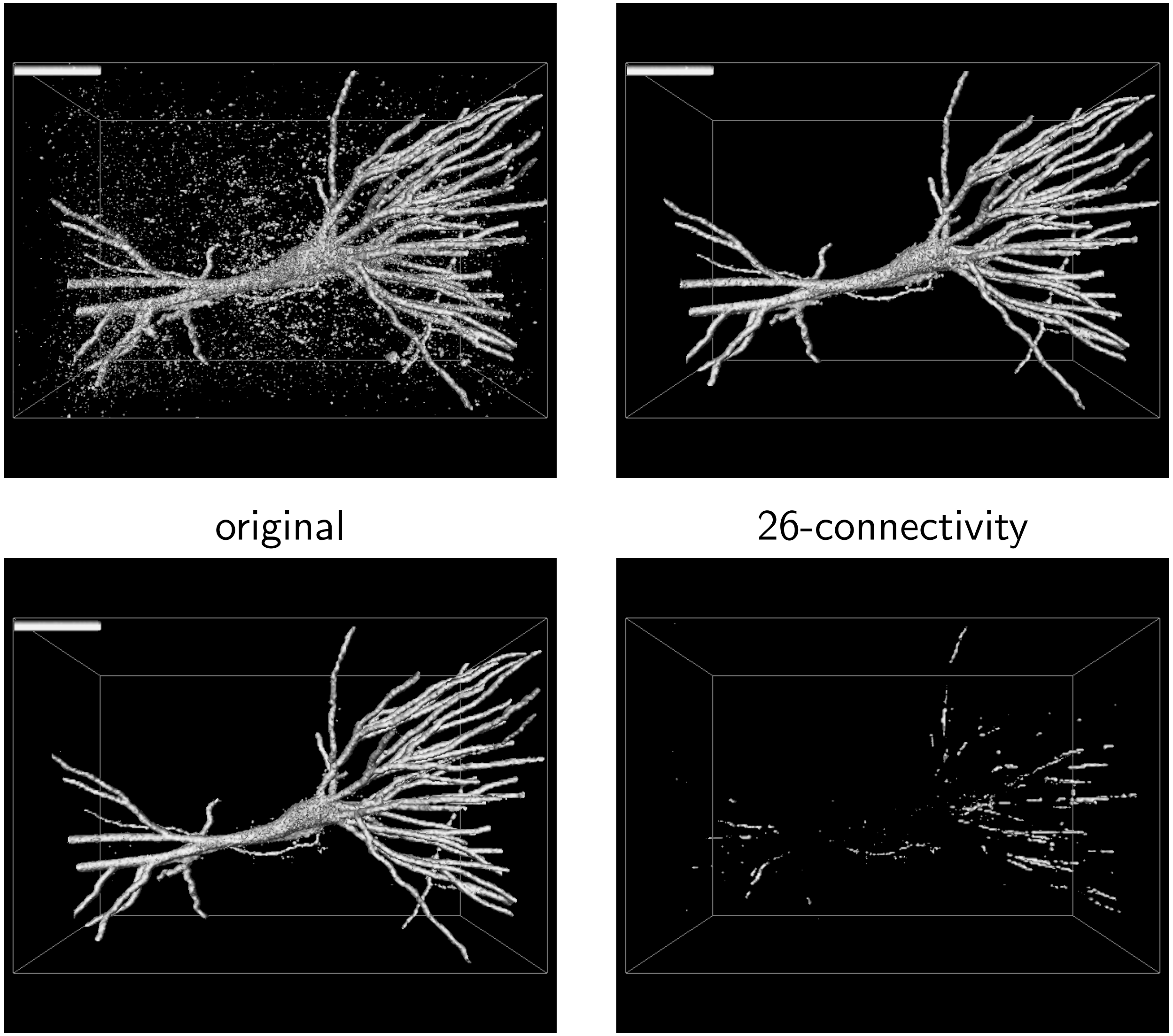


$X$ $\psi(X)$ $\Gamma_p^{\psi}(X)$ $\Gamma_q^{\psi}(X)$

**Partitioning-based connectivity**

Note that $p = (65, 85)$ and $q = (200, 225)$.

# Second-Generation Connected Filtering in 3-D



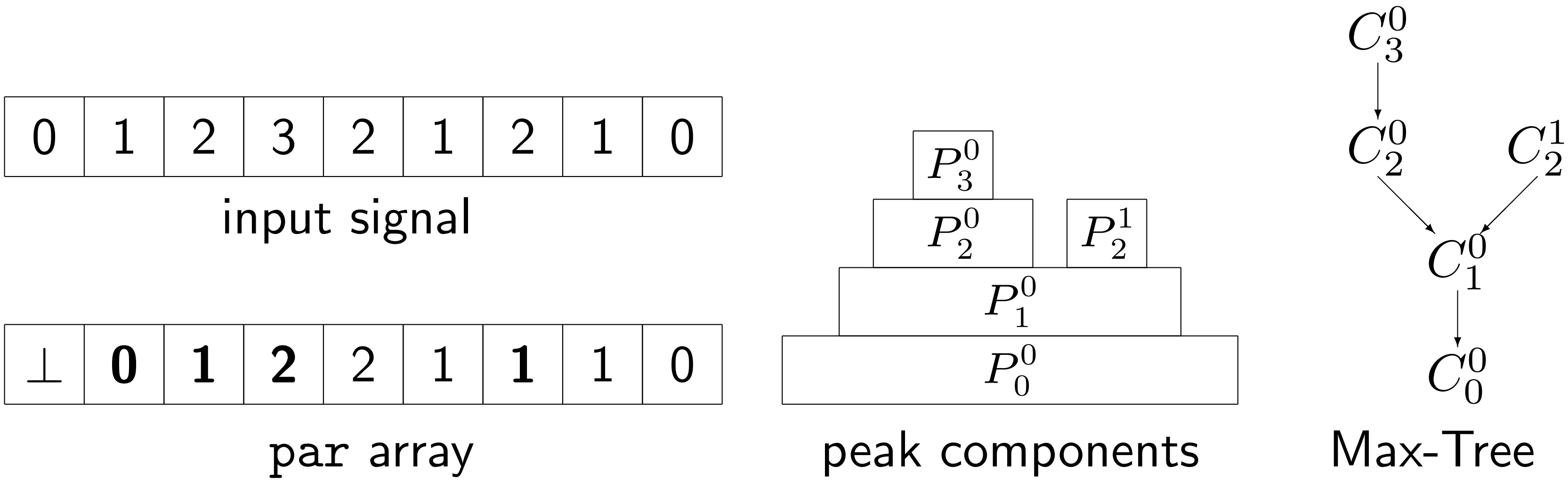
original

26-connectivity

closing based connectivity

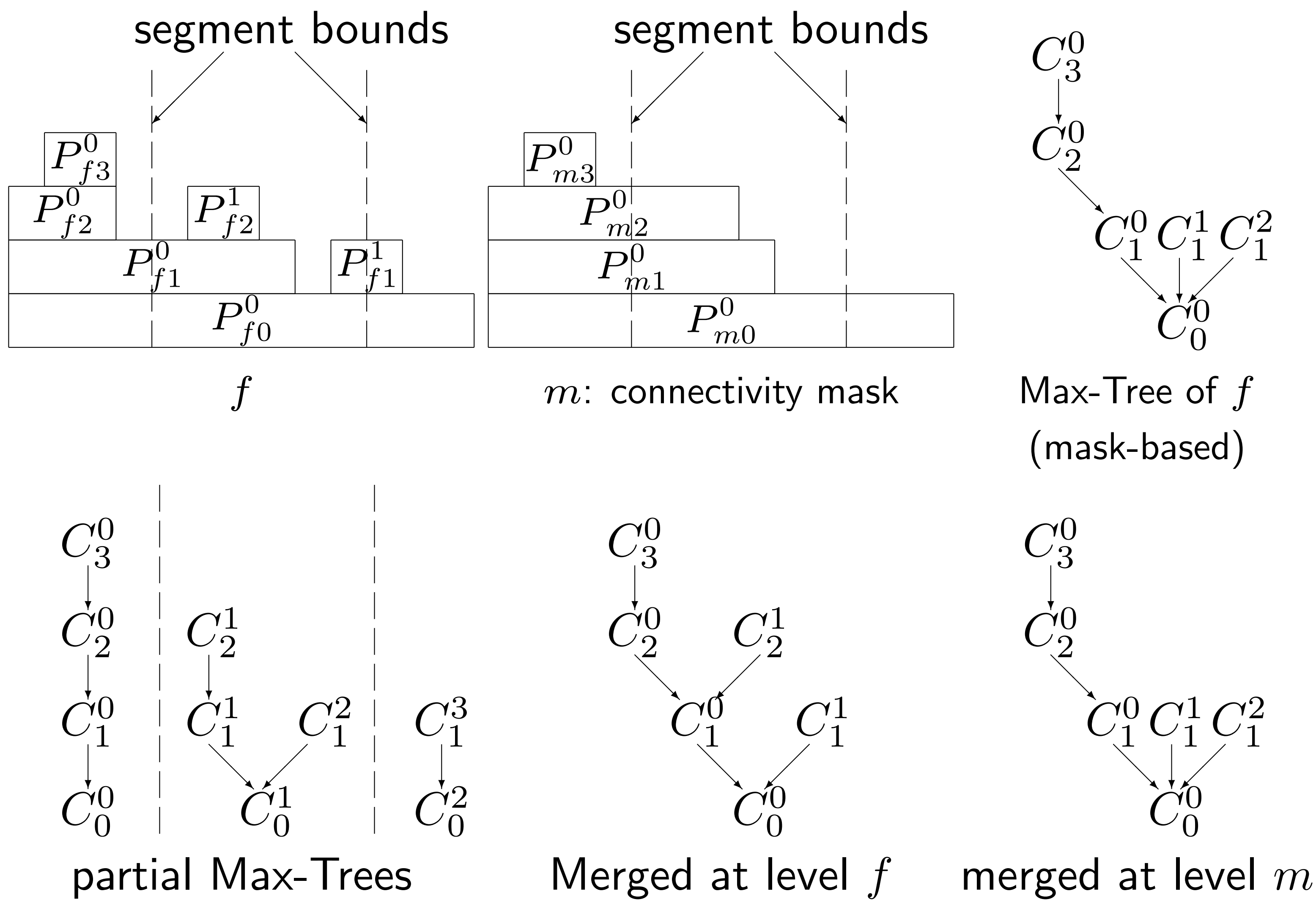
difference

# Including union-find in the Max-Tree




- Example of input signal, peak components, Max-Tree and its encoding in a `par` array.
- $\perp$ denotes the overall root node, and boldface numbers denote the level roots, i.e., they point to positions in the input with grey level other than their own.

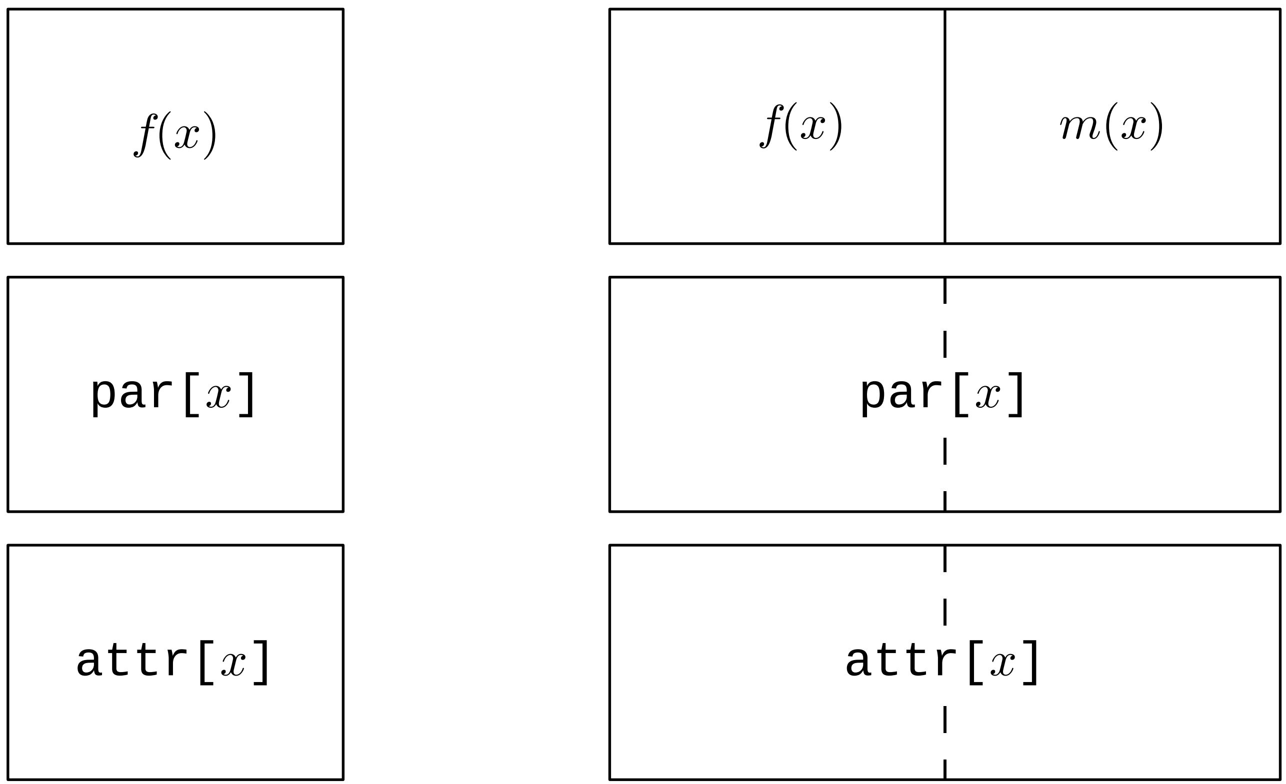
# Merging in Dual-Input Max-Trees



# Adaptations to the Parallel Max-Tree Algorithm

- We store $f$ and $m$ consecutively in a single array of length $2 * volsize$ and maintain an array `levroot` of length $G$ for each thread.
- Whenever $m(x) = f(x)$ the algorithm does not change fundamentally
- Whenever $m(x) \neq f(x)$ we check whether there is a level root at level $h = m(x)$, and at $f(x)$
- If not, `levroot`$[m(x)] \leftarrow x + volsize$ and/or `levroot`$[f(x)] \leftarrow x$
- We then set `par`$[x]$ to `levroot`$[f(x)]$ and `par`$[x + volsize]$ to `levroot`$[m(x)]$
- If when flooding we find a pixel at level $h$ with $f(y) = m(x)$, and `levroot`$[h] = x + volsize$ we set `levroot`$[h]$ and `par`$[x + volsize]$ to $y$
- If $m(x) < f(x)$ we immediately finalize the node at level $f(x)$ and set `levroot`$[f(x)]$ to $\perp$.

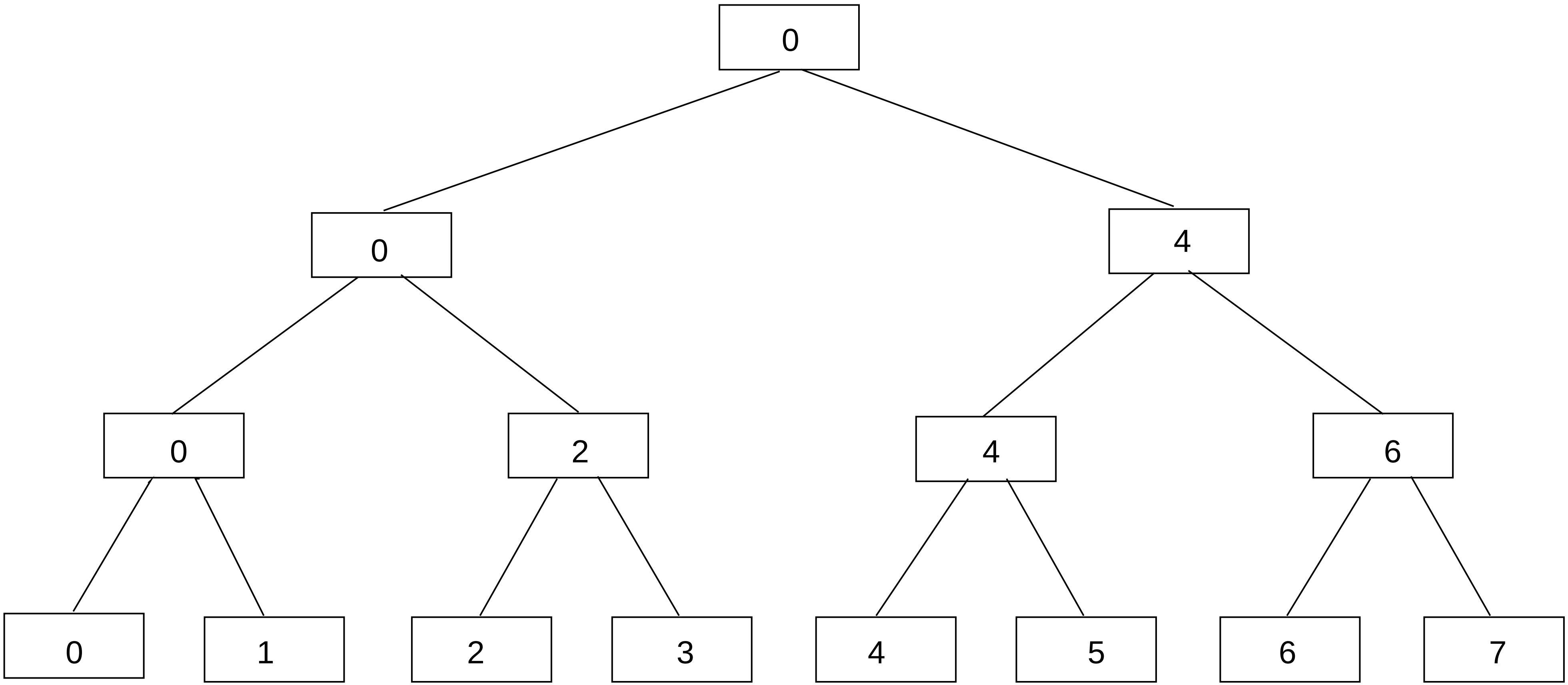
# Data Structures




Standard Parallel

Dual-Input Parallel

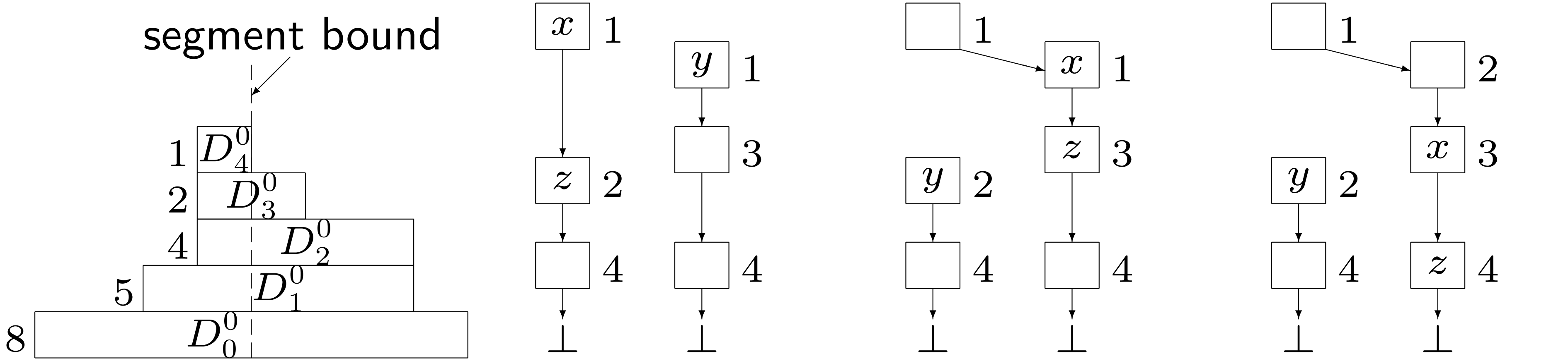
# Binary Tree used for Merging Domains



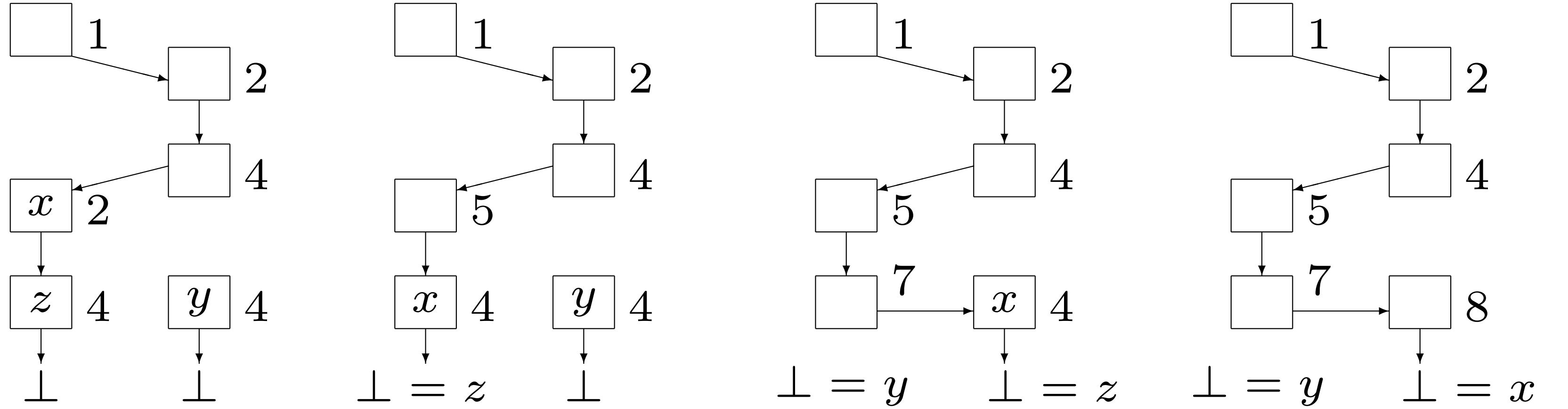
# Concurrent construction and filtering thread $p$.

```
process ccaf(p)
    build dual input Max-Tree Tree(p) for segment belonging to p
    var i := 1 , q := p ;
    while p + i < N_p ∧ q mod 2 = 0 do
        wait to glue with right-hand neighbor ;
        for all edges (x, y) between Tree(p) and Tree(p + i) do
            if f(x) ≠ m(x) then x := x + volsize;
            if f(y) ≠ m(y) then y := y + volsize;
            connect(x, y) ;
        end ;
        i := 2 * i ; q := q/2 ;
    end ;
    if p = 0 then
        release the waiting threads
    else
        signal left-hand neighbor ;
        wait for thread 0
    end ;
    filter(p, lambda) ;
end ccaf.
```

# Merging Two Max-Trees

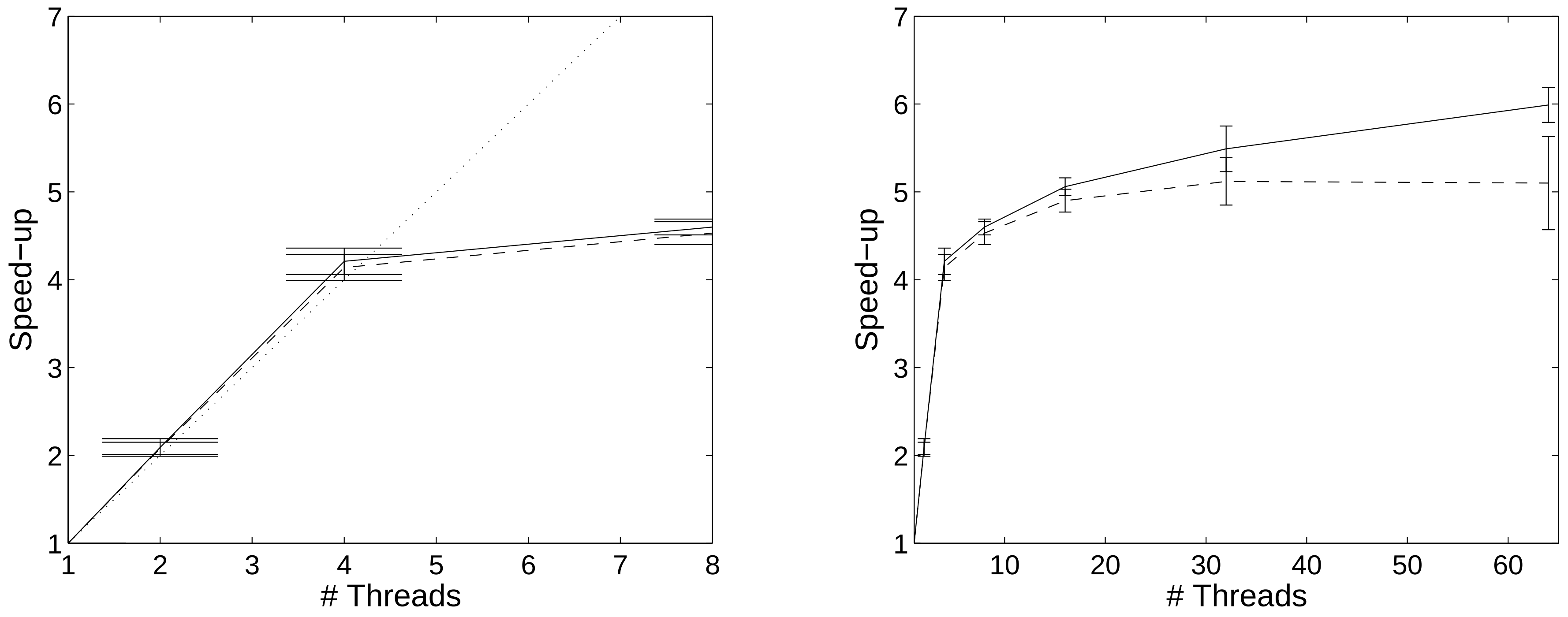



(a) signal (b) start; cor=0 (c) step 1; cor=1 (d) step 2; cor=1



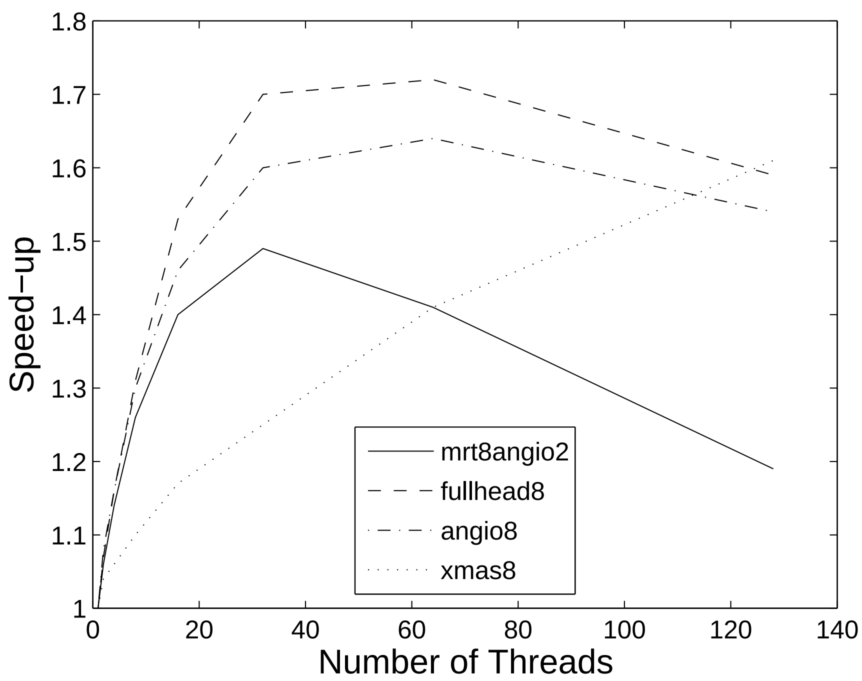
(e) step 3; cor= 3 (f) step 4; cor= 3 (g) step 5; cor=4 (h) final tree

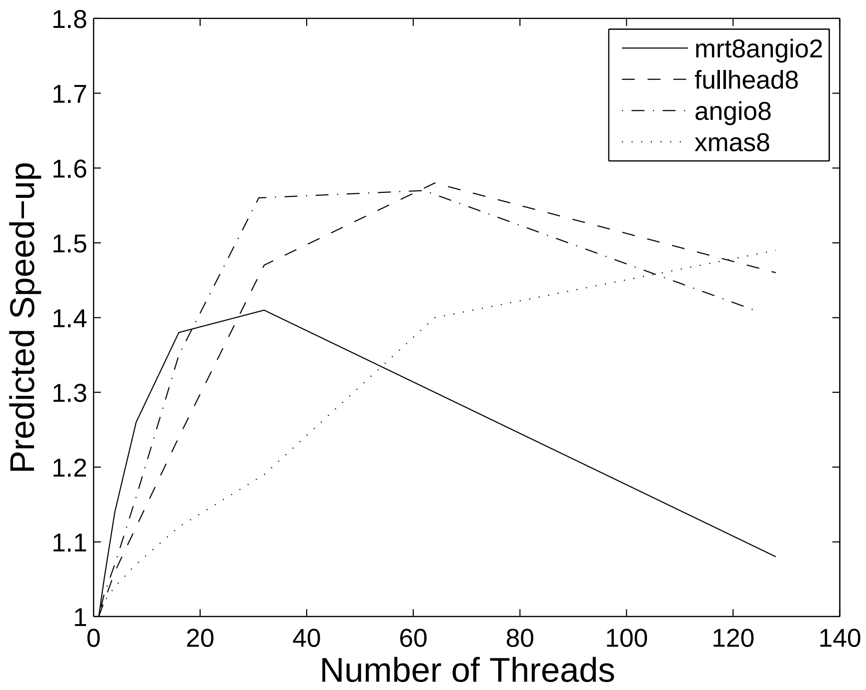
# Timings




- Speed-up for volume openings (solid) and non-compactness thinnings (dashed) as a function of number of threads.
- Timings were performed on a 2-socket, dual-core opteron-based workstation (4 core total).

# Speed-up for $N_{threads} > N_p$




**Left:** Speed-up on single-core CPU (P4-520, 3.0 GHz)

**Right:** Predicted perfomance from cache thrashing (using valgrind)

# Complexity

- Assuming a volume of $X \times Y \times Z = N$, in the building phase the time complexity is $O(GN/N_p)$
- This complexity arises from the $O(GN)$ complexity of Salembier et al's Max-Tree algorithm
- If the number of grey levels is large, it may be better to replace this by Najman and Couprie's method.
- The merging phase has complexity $O(GXY \log N \log N_p)$ if the volume is split up into slices orthogonal to the $Z$ direction.
- The $\log N$ is due to the fact that we only use path compression, not union-by-rank.
- Memory requirements are $O(N + G)$.

university of groningen

# Conclusions

- As with the regular Max-Tree algorithm, the Dual-Input Max-Tree algorithm is parallellizable through the use of union-find.
- Speed-up is (slightly better than) linear over four cores.
- Tests using more threads than cores suggest that improved speed-up beyond 4 CPUs is expected.
- This extended speed-up for $N_{\text{threads}} > N_p$ is due to reduced cache thrashing.
- For very high numbers of grey levels in particular, we should replace the building phase by a version of Najman and Courpie's algorithm (IEEE Trans. Image Proc. 2006).

# Questions

?