Component-hypertrees for image segmentation

Nicolas Passat[1], Benoît Naegel[2]

[1]LSIIT, UMR 7005, Strasbourg University, France
[2]LORIA, UMR 7503, Nancy University, France

ISMM'11
Intra, Italia

Outline

1. Introduction
2. Component-hypertrees
3. Component-hypertrees and segmentation – A case study
4. Experiments
5. Conclusion

Outline

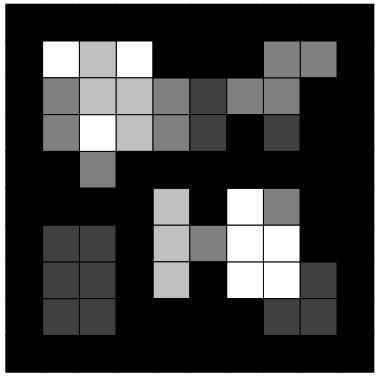
1. Introduction
2. Component-hypertrees
3. Component-hypertrees and segmentation – A case study
4. Experiments
5. Conclusion

1.1 Component-trees

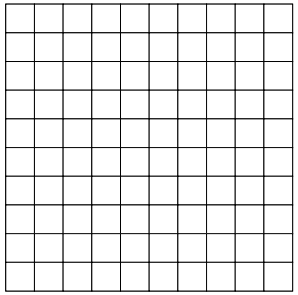
Component-trees: Definition

- $F : E \to V$ a grey-level image
- Threshold set $X_v(F) \subseteq E$ of F at level $v \in V$

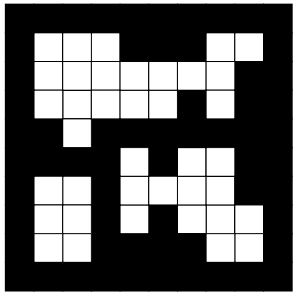
$$X_v(F) = \{x \in E \mid v \leq F(x)\}$$



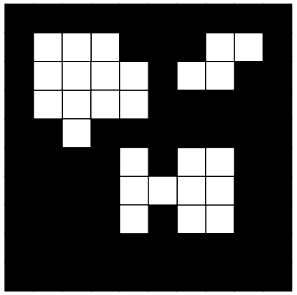
F



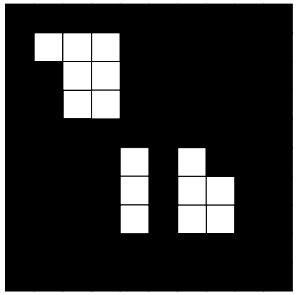
$X_0(F)$



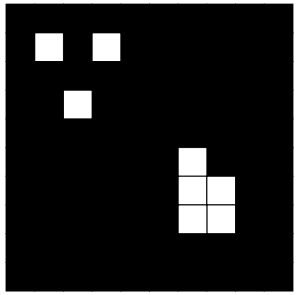
$X_1(F)$



$X_2(F)$



$X_3(F)$

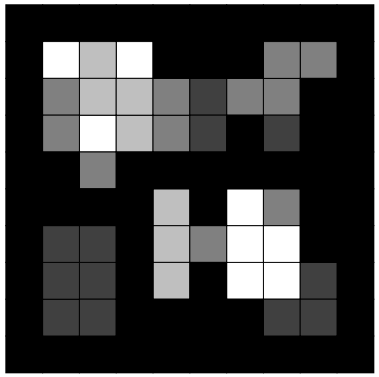


$X_4(F)$

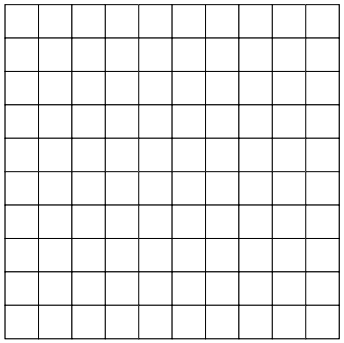
1.1 Component-trees

Component-trees: Definition

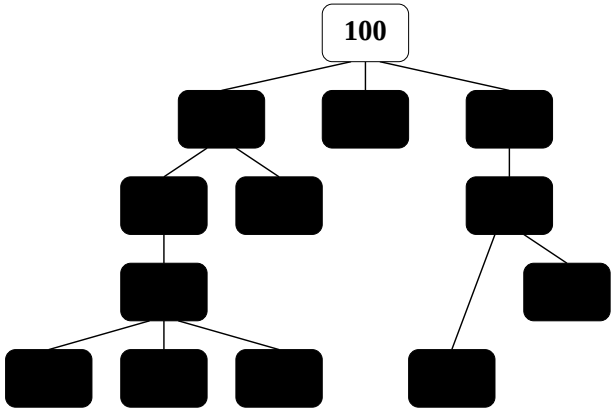
- $\mathcal{C}[X_v(F)]$ connected components of $X_v(F)$
- $\mathcal{K} = \bigcup_{v \in V} \mathcal{C}[X_v(F)]$
  connected components of all threshold sets
- $(\mathcal{K}, \subseteq)$ is a partially-ordered set



F



$X_0(F)$

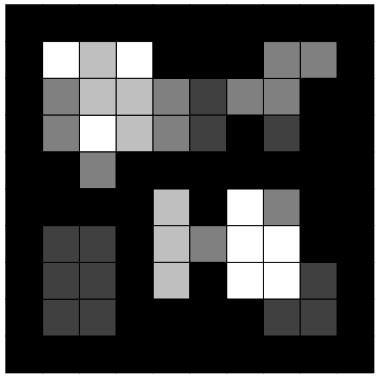


Component-tree of F
(with an *area* attribute)

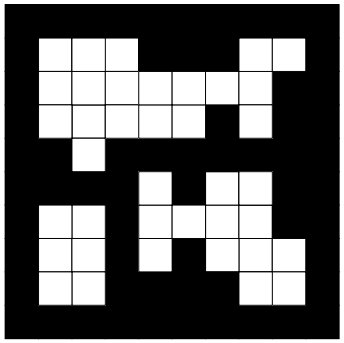
1.1 Component-trees

Component-tree: Definition

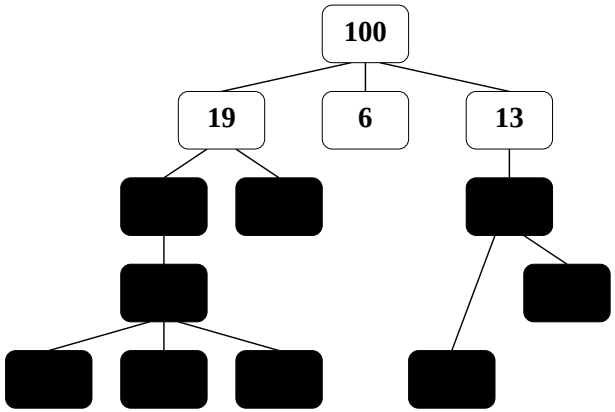
- $\mathcal{C}[X_v(F)]$ connected components of $X_v(F)$
- $\mathcal{K} = \bigcup_{v \in V} \mathcal{C}[X_v(F)]$
  connected components of all threshold sets
- $(\mathcal{K}, \subseteq)$ is a partially-ordered set



F



$X_1(F)$

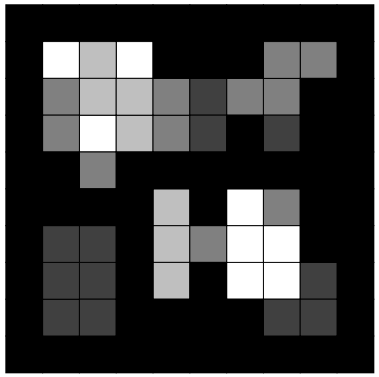


Component-tree of F
(with an *area* attribute)

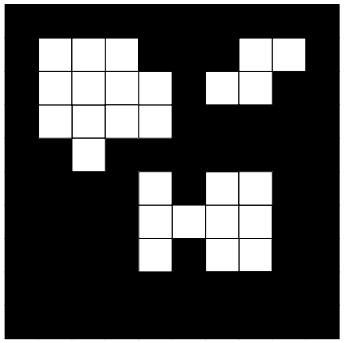
1.1 Component-trees

Component-tree: Definition

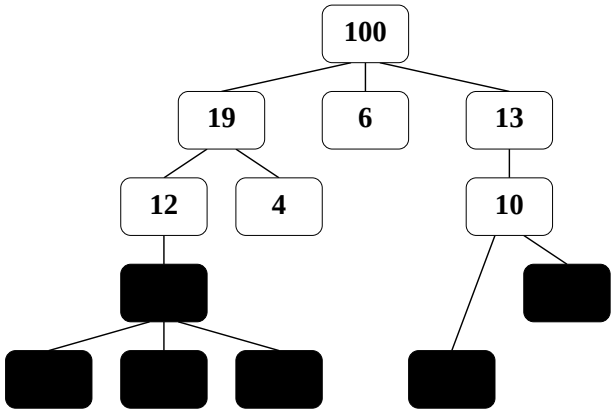
- $\mathcal{C}[X_v(F)]$ connected components of $X_v(F)$
- $\mathcal{K} = \bigcup_{v \in V} \mathcal{C}[X_v(F)]$
  connected components of all threshold sets
- $(\mathcal{K}, \subseteq)$ is a partially-ordered set



F



$X_2(F)$

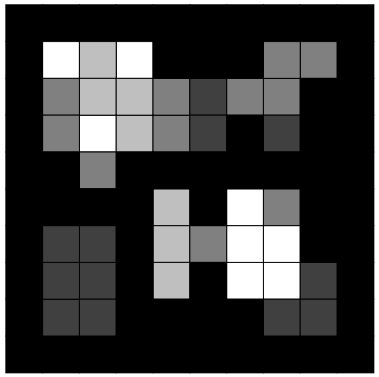


Component-tree of F
(with an *area* attribute)

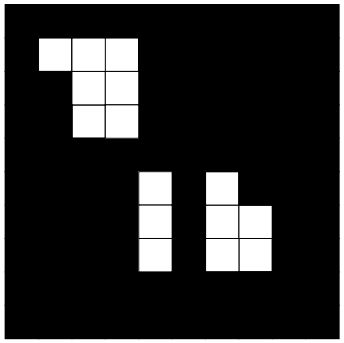
1.1 Component-trees

Component-tree: Definition

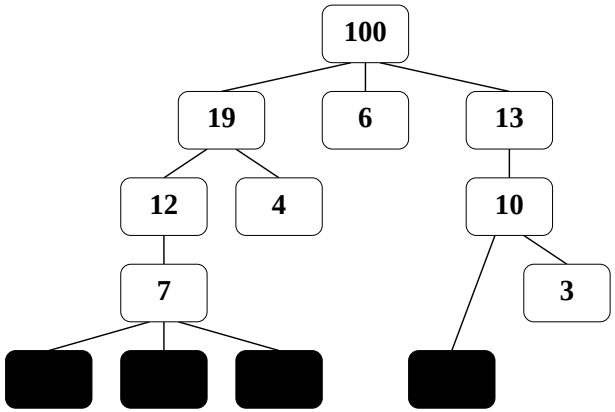
- $\mathcal{C}[X_v(F)]$ connected components of $X_v(F)$
- $\mathcal{K} = \bigcup_{v \in V} \mathcal{C}[X_v(F)]$
  connected components of all threshold sets
- $(\mathcal{K}, \subseteq)$ is a partially-ordered set



F



$X_3(F)$

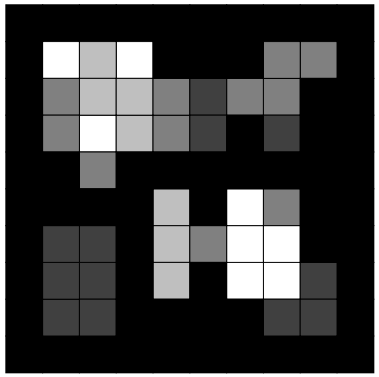


Component-tree of F
(with an *area* attribute)

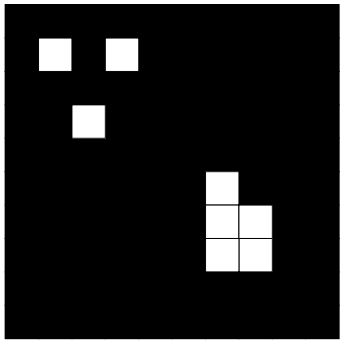
1.1 Component-trees

Component-tree: Definition

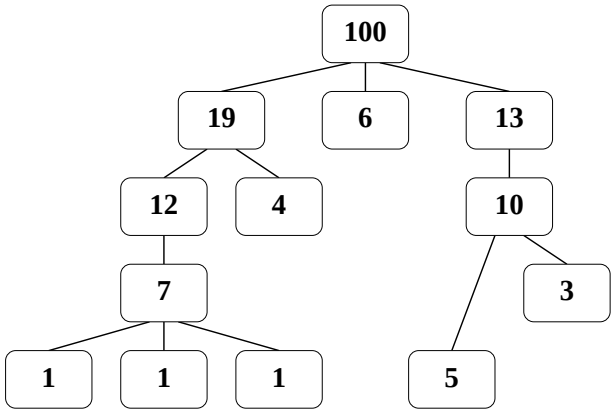
- $\mathcal{C}[X_v(F)]$ connected components of $X_v(F)$
- $\mathcal{K} = \bigcup_{v \in V} \mathcal{C}[X_v(F)]$
  connected components of all threshold sets
- $(\mathcal{K}, \subseteq)$ is a partially-ordered set



F



$X_4(F)$

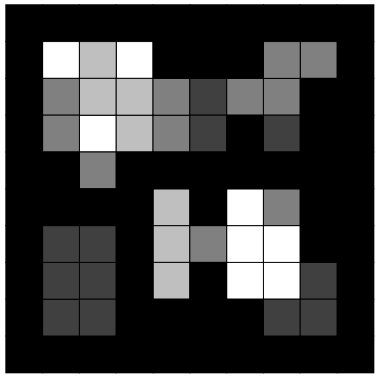


Component-tree of F
(with an *area* attribute)

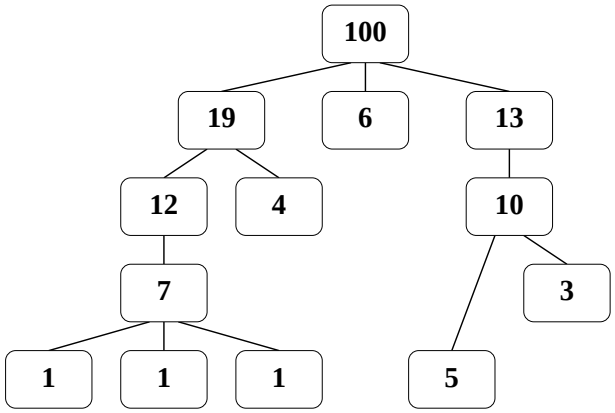
1.1 Component-trees

Component-tree: Definition

- Comp.-tree of $F \sim$ Hasse diagram of $(\mathcal{K}, \subseteq)$
- Nodes $\sim$ Connected components of $\mathcal{K}$
- Root $\sim \sup(\mathcal{K}, \subseteq) = E$



F



Component-tree of F
(with an *area* attribute)

1.1 Component-trees

Component-trees and image processing (non exhaustive)

- Connected filters (Breen, 1996), (Salembier, 1998)
- Segmentation (Jones, 1999)
- Registration (Mattes, 1999)
- Image retrieval (Mosorov, 2005)
- Binarisation (Naegel, 2007)
- Visualization (Westenberg, 2007)
- Classification (Urbach, 2007)
- etc.

1.2 Connectivity

"Standard" connectivities

In discrete (binary) images

$$adjacency \rightarrow connectivity$$

For a binary image $B \subseteq E$ (E finite) and a (symmetric, irreflexive) relation $\frown$ on E

- $x, y \in B$ are adjacent if $x \frown y$
- $x, y \in B$ are connected if there exists a path $(a_i)_i$ in B from x to y :
  $x = a_0 \frown a_1 \frown \ldots \frown a_{k-1} \frown a_k = y$
- $C \subseteq B$ is a connected set if all its elements are connected
- a connected component C of B is a maximal connected set

Examples in $\mathbb{Z}^n$

- $2n$-connectivity (strong connectivity): 4-connectivity in $\mathbb{Z}^2$, 6-connectivity in $\mathbb{Z}^3$
- (3^n-1)-connectivity (weak connectivity): 8-connectivity in $\mathbb{Z}^2$, 26-connectivity in $\mathbb{Z}^3$

1.2 Connectivity

"Standard connectivities": based on an equivalence relation

$$\begin{tikzcd}
\textit{Connectedness} \arrow[r] \arrow[d] & \textit{Connected components} \\
\textit{Equivalence relation} \arrow[r] & \textit{Equivalence classes} \arrow[u]
\end{tikzcd}$$

An alternative definition of connectivities (based on a partition)

$$\begin{tikzcd}
\textit{Connectedness} & \textit{Connected components} \arrow[l] \arrow[d] \\
\textit{Equivalence relation} \arrow[u] & \textit{Equivalence classes} \arrow[l]
\end{tikzcd}$$

Compliant with the definition of morphological connectivities (Ronse, Serra, 1998)

1.2 Connectivity

Morphological connectivity and connectivity hierarchies

- Morphological connectivity on $B \subseteq E$ $\Leftrightarrow$ Definition of a partition of B
- Possibility to define a hierarchy of partitions of B ($\sim$ n-ary partition-tree)

$\Rightarrow$ Possibility to define a hierarchy of connectivities

- from the strongest one: B is the only connected component
- to the weakest one: $\{x\}$ is a connected component for all $x \in B$

Example: (a subcase of) mask-based connectivity (Ouzounis, 2007)

$B \subseteq E$: a binary image, equipped with a connectivity
$\omega(B) \subseteq E$: a mask of B

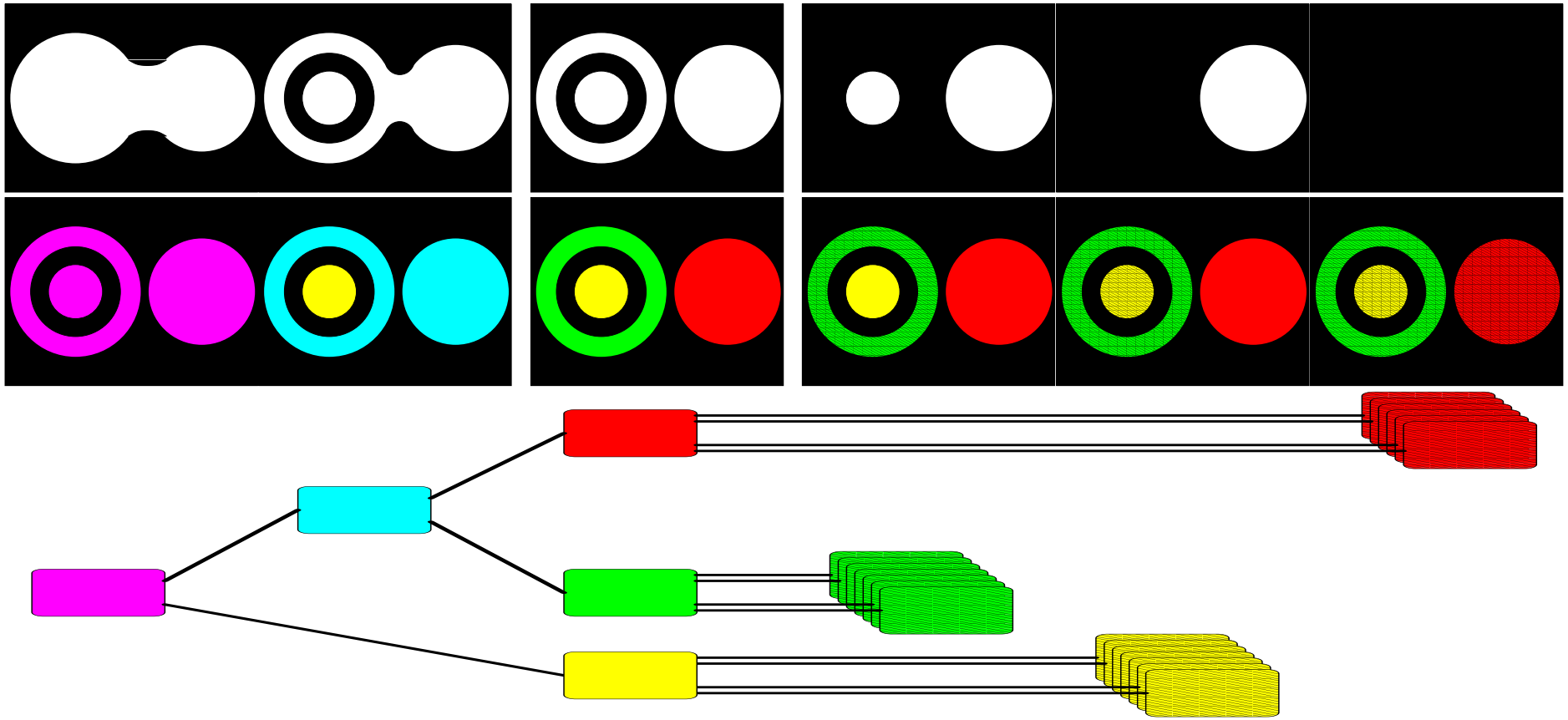
- If $\omega(B) \subseteq B$, the ω-connected components of B are
  - the connected components of $\omega(B)$
  - $\{x\}$ for all $x \in B \setminus \omega(B)$
- If $\omega(B) \supseteq B$, the ω-connected components of B are
  - the sets $B \cap X$ for all the connected component X of $\omega(B)$

1.2 Connectivity

Example: mask-based connectivity (contd.)

Relation $\subseteq$ on the masks $\omega(B) \rightarrow$ Hierarchy between the mask-based ω-connectivities
Ex:

$$E \supseteq \ldots \supseteq \varepsilon^k \circ \delta^k(B) \supseteq \ldots \supseteq \varepsilon \circ \delta(B) \supseteq B \supseteq \delta \circ \varepsilon(B) \supseteq \ldots \supseteq \delta^k \circ \varepsilon^k(B) \supseteq \ldots \supseteq \emptyset$$



1.3 Purpose

Motivation

- A component-tree is defined “only” for a given connectivity
- Several (hierarchical) connectivities can be used to analyse an image in a “topological multiscale” fashion

Purpose

- Enrich the component-trees with a hierarchy of connectivities
  $\rightarrow$ Notion of component-hypertree
- Illustrate the use of component-hypertrees for image segmentation
  $\rightarrow$ Extension of an example-based segmentation method relying on component-trees

Outline

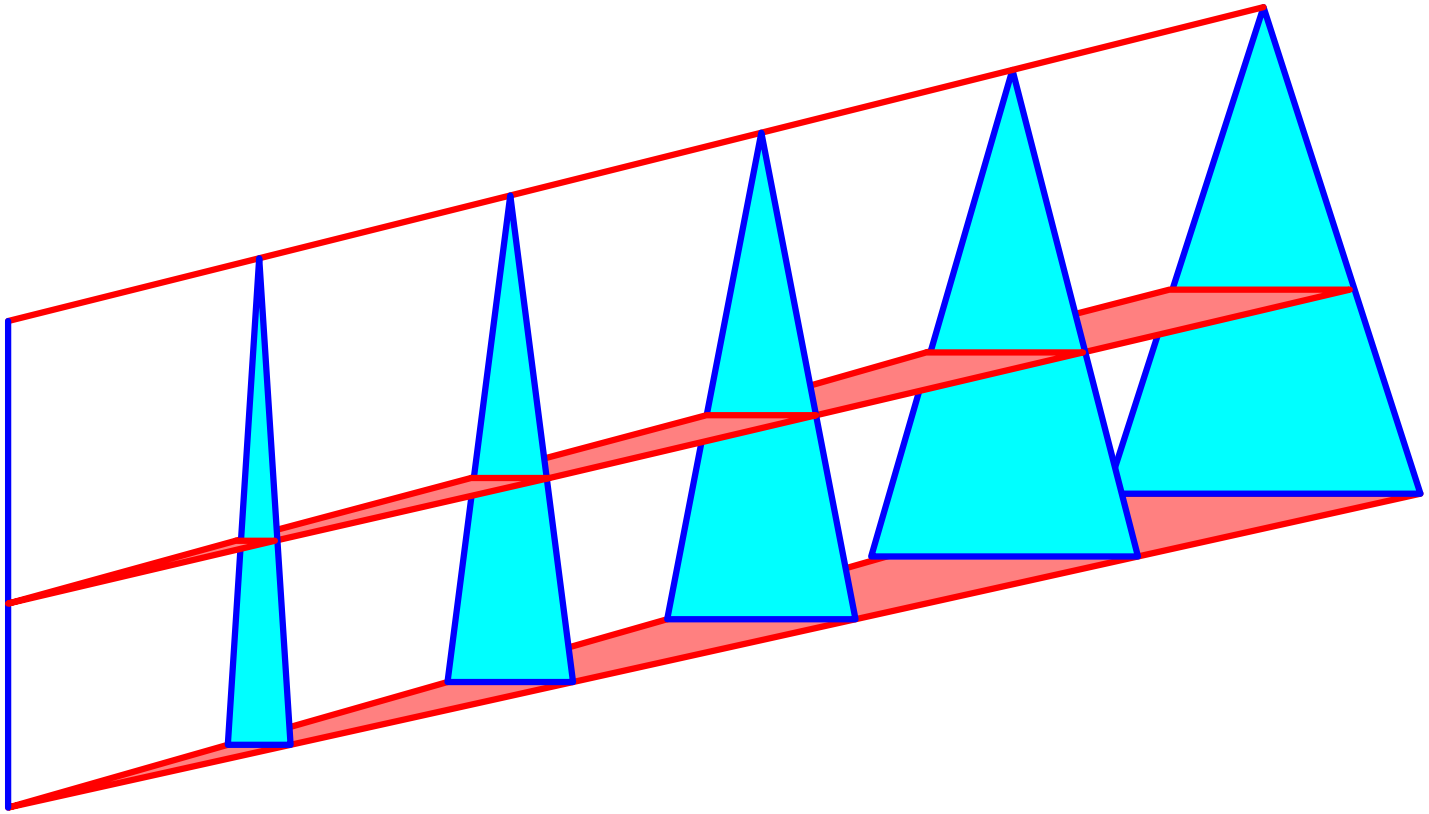
1. Introduction
2. Component-hypertrees
3. Component-hypertrees and segmentation – A case study
4. Experiments
5. Conclusion

2.1 Component-hypertrees: Definition

Component-hypertrees: Informal definition

Data structure composed of

- ($\triangle$) a component-tree of $I : E \rightarrow V$ for each connectivity C
- ($\triangle$) a partition-tree for each threshold image $X_v(I)$, $v \in V$



2.1 Component-hypertrees: Definition

Component-hypertrees: Formal definition

Image: $I : \mathbb{Z}^n \to V$
Masks: $\inf V = \Omega_t(I) \leq \ldots \leq \Omega_{-1}(I) \leq \Omega_0(I) \leq \Omega_1(I) \leq \ldots \leq \Omega_u(I) = \sup V$

$(\mathcal{K}_i, L_i, R_i)$: the (mask-based connectivity) Ω_i-component tree of I

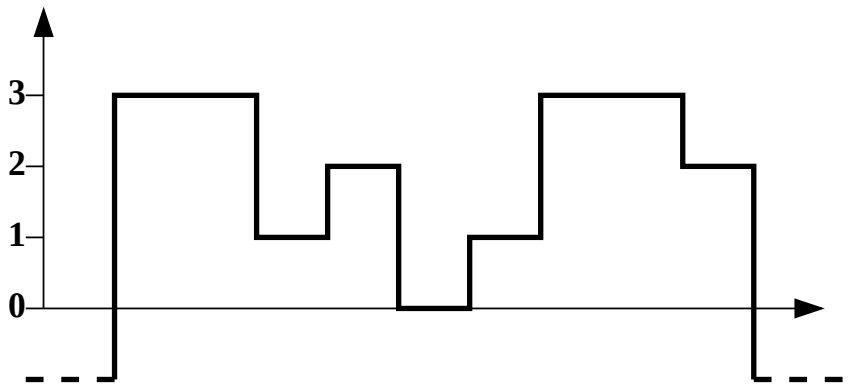
The component-hypertree of I is $H = (\mathcal{H}, L_\downarrow, L_\rightarrow)$ with

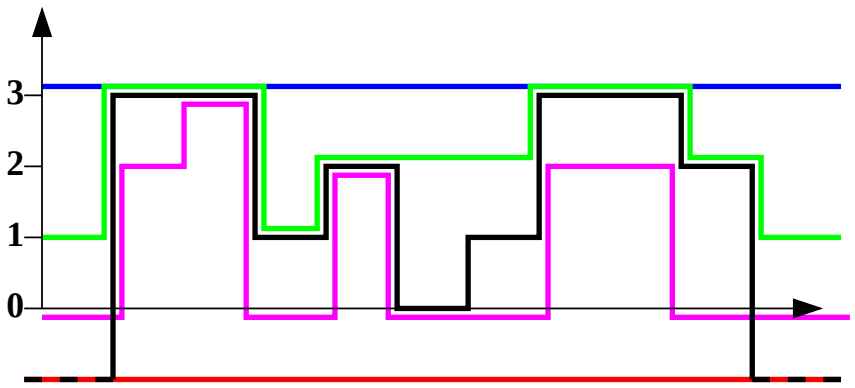
- $\mathcal{H} = \bigcup_{i=t}^{u} \mathcal{K}_i$
- $\forall i \in [\![t, u]\!]$, the subgraph of $(\mathcal{H}, L_\downarrow)$ induced by $\mathcal{K}_i \subseteq \mathcal{H}$ is the Ω_i-component-tree of I
- $\forall v \in V$, the subgraph of $(\mathcal{H}, L_\rightarrow)$ induced by $\mathcal{S}_v = \bigcup_{i \in [\![t,u]\!]} \mathcal{C}_{\omega_i}[X_v(I)] \subseteq \mathcal{H}$ is the partition-tree of $X_v(I)$, i.e. the Hasse diagram of $(\mathcal{S}_v, \subseteq)$

2.2 Component-hypertrees: Illustration

Component-hypertrees: Illustration

- A 1D image $I : \mathbb{Z} \rightarrow [0, 3] \cup \{-\infty\}$
- 5 masks $\Omega_i(I) \rightarrow$ 5 connectivities

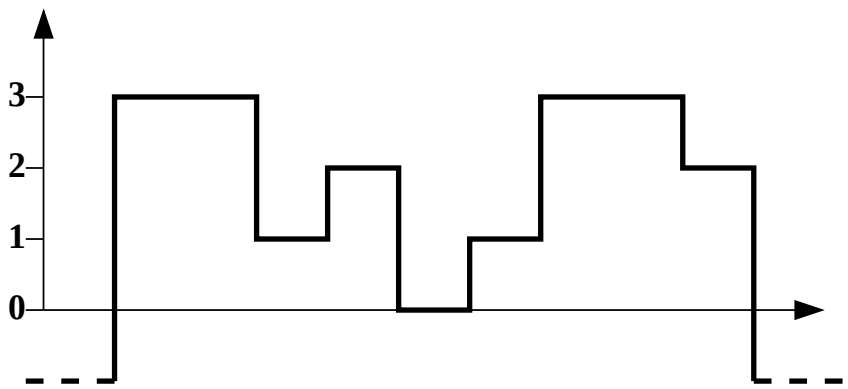


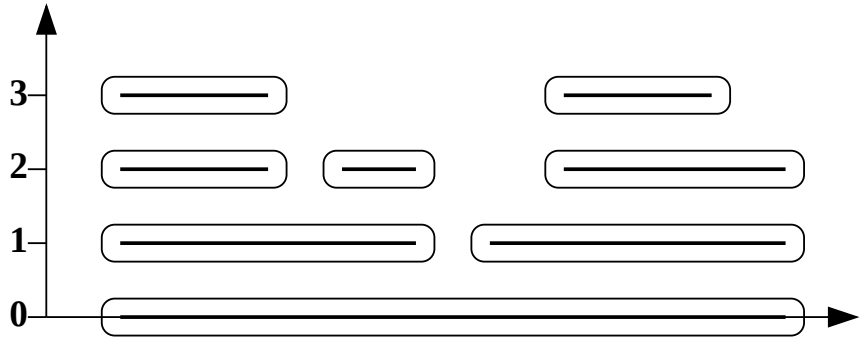


2.2 Component-hypertrees: Illustration

Component-hypertrees: Illustration

- A 1D image $I : \mathbb{Z} \rightarrow [0, 3] \cup \{-\infty\}$
- 5 masks $\Omega_i(I) \rightarrow$ 5 connectivities

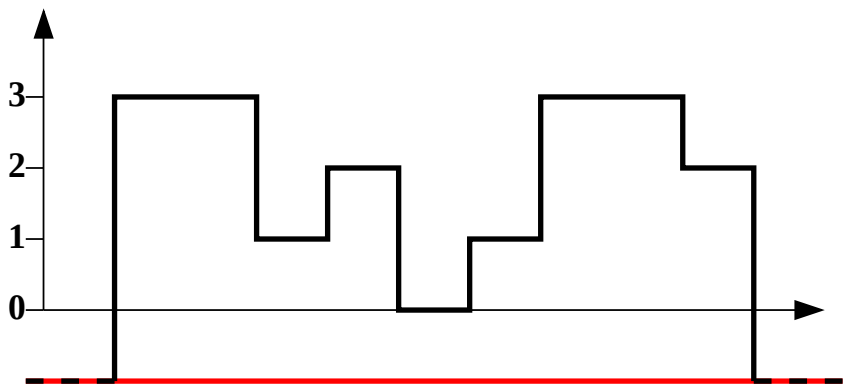


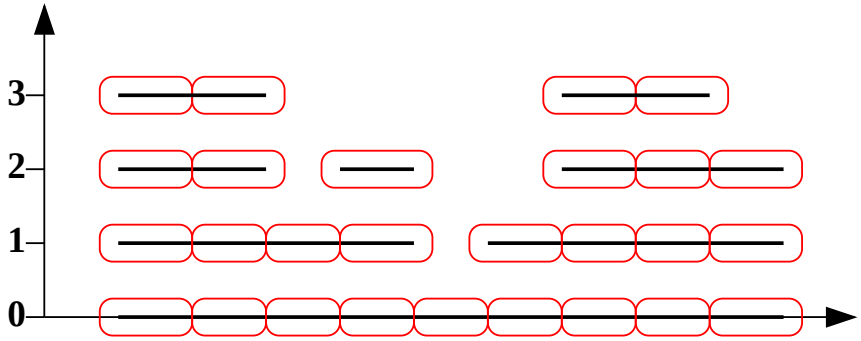


2.2 Component-hypertrees: Illustration

Component-hypertrees: Illustration

- A 1D image $I : \mathbb{Z} \rightarrow [0, 3] \cup \{-\infty\}$
- 5 masks $\Omega_i(I) \rightarrow$ 5 connectivities

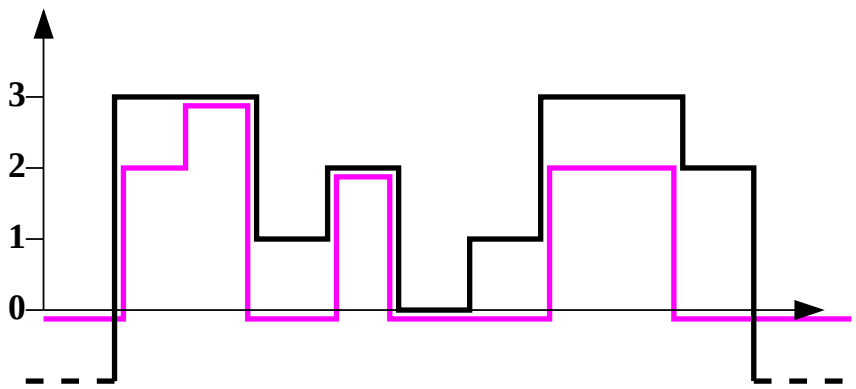


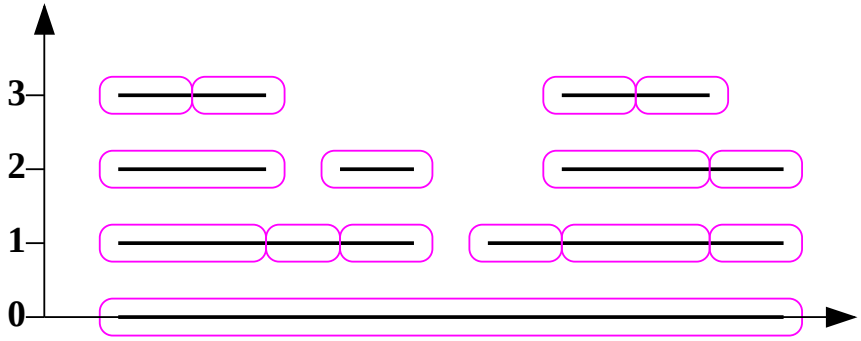


2.2 Component-hypertrees: Illustration

Component-hypertrees: Illustration

- A 1D image $I : \mathbb{Z} \rightarrow [0, 3] \cup \{-\infty\}$
- 5 masks $\Omega_i(I) \rightarrow$ 5 connectivities

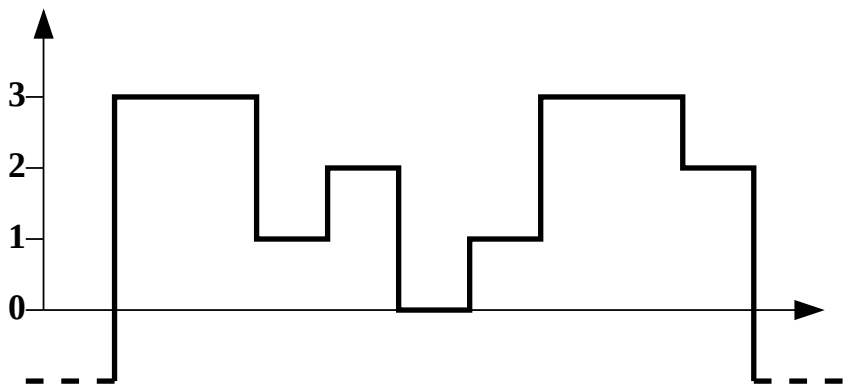


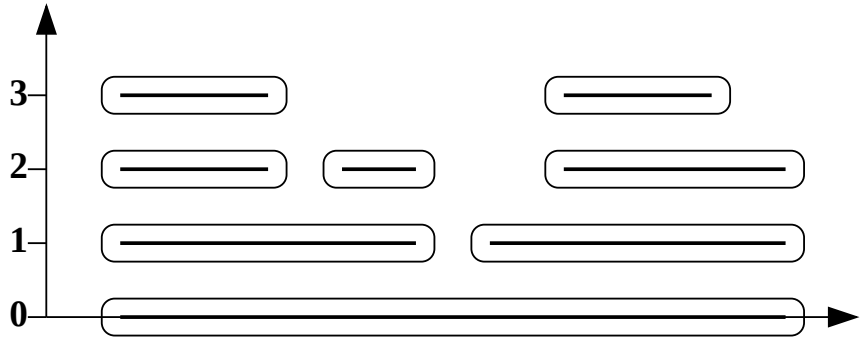


2.2 Component-hypertrees: Illustration

Component-hypertrees: Illustration

- A 1D image $I : \mathbb{Z} \rightarrow [0, 3] \cup \{-\infty\}$
- 5 masks $\Omega_i(I) \rightarrow$ 5 connectivities

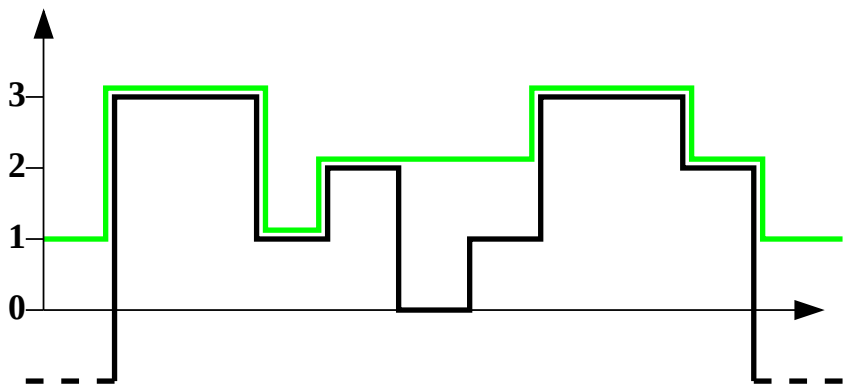


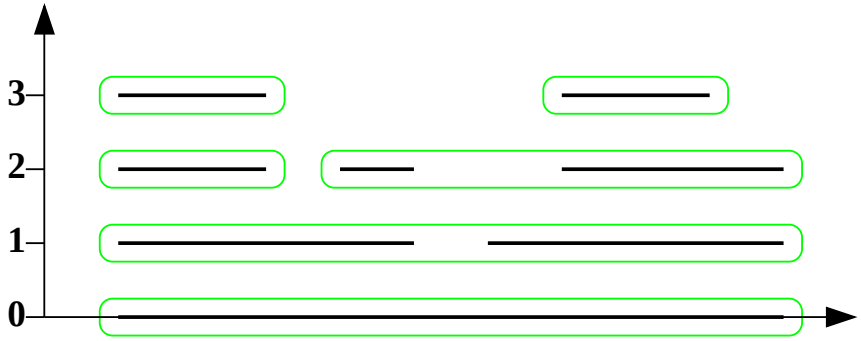


2.2 Component-hypertrees: Illustration

Component-hypertrees: Illustration

- A 1D image $I : \mathbb{Z} \rightarrow [0, 3] \cup \{-\infty\}$
- 5 masks $\Omega_i(I) \rightarrow$ 5 connectivities

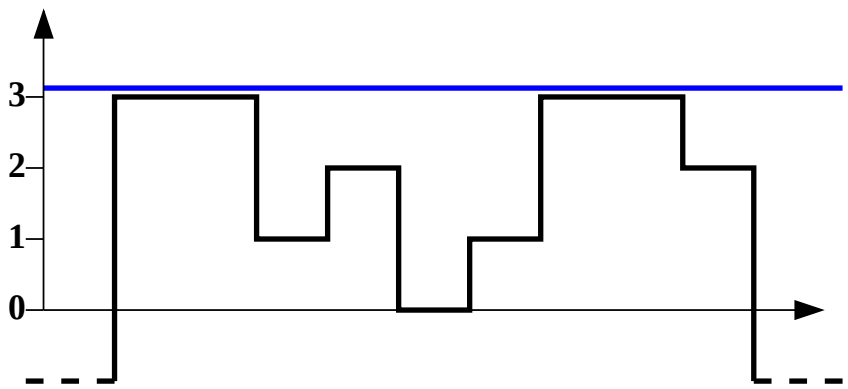


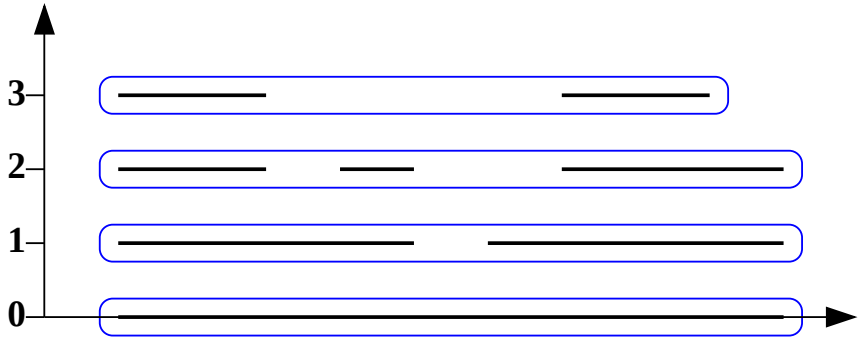


2.2 Component-hypertrees: Illustration

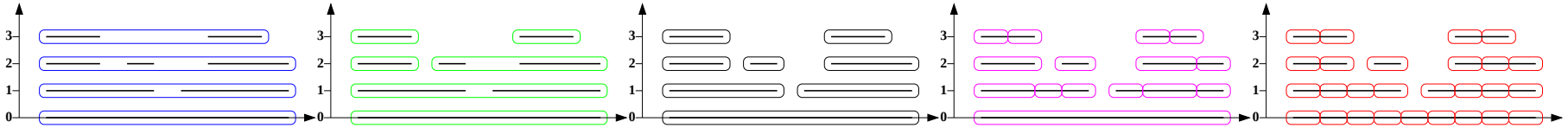
Component-hypertrees: Illustration

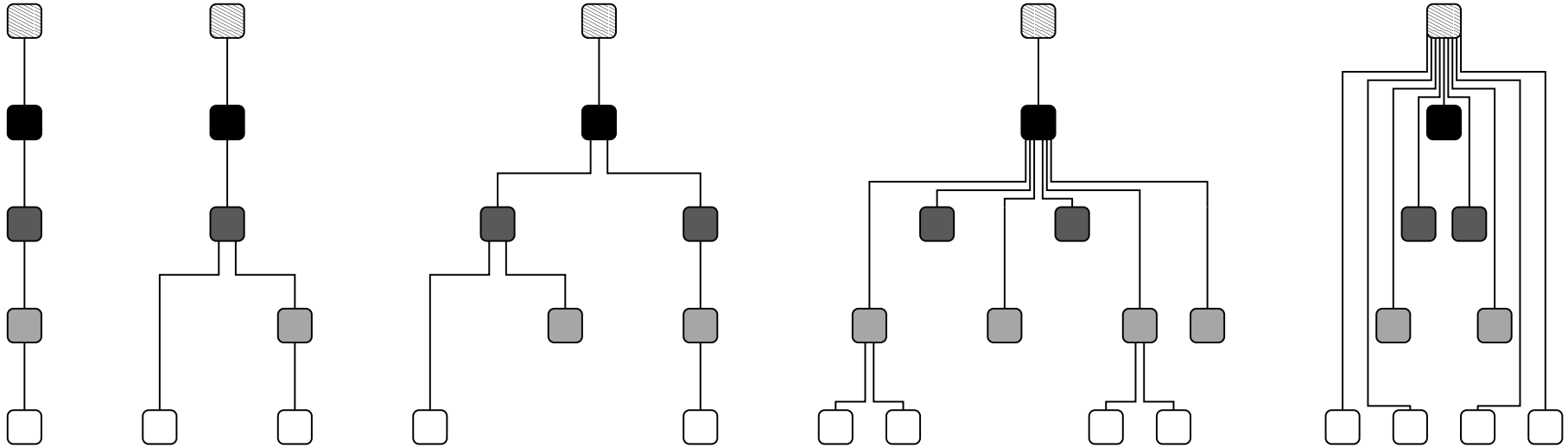
- A 1D image $I : \mathbb{Z} \rightarrow [0, 3] \cup \{-\infty\}$
- 5 masks $\Omega_i(I) \rightarrow$ 5 connectivities



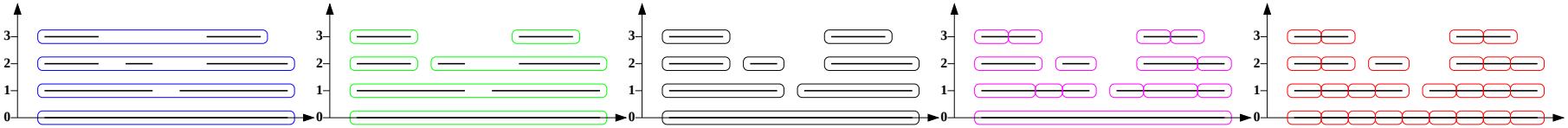


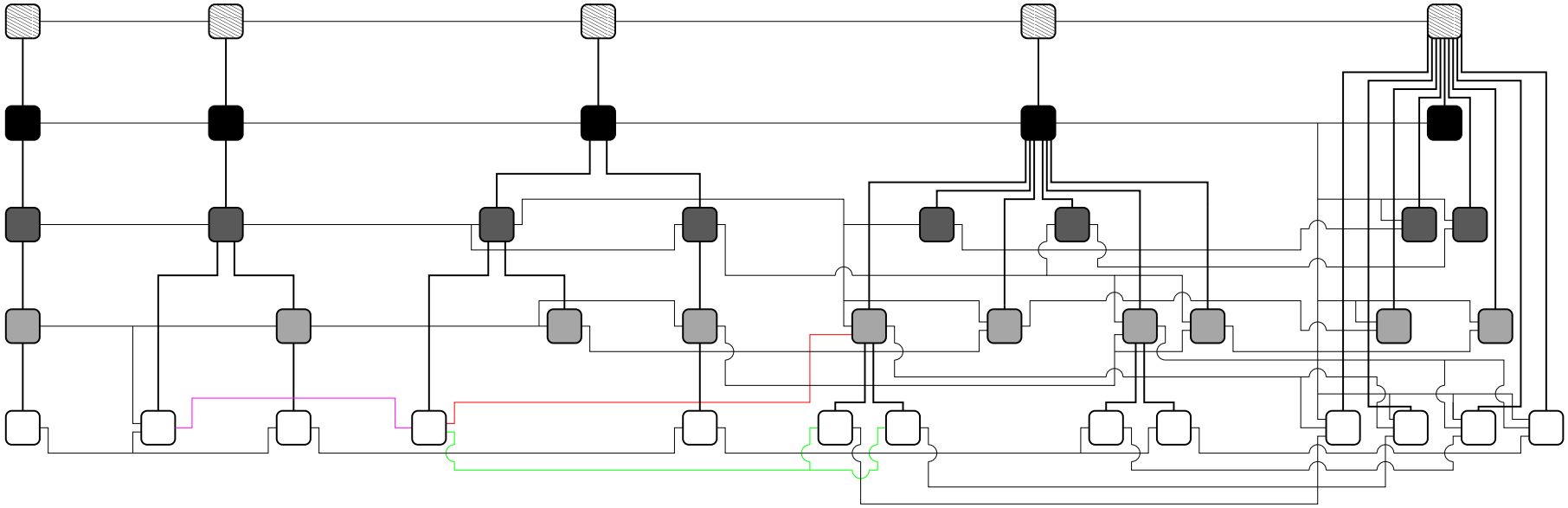
2.2 Component-hypertrees: Illustration





2.2 Component-hypertrees: Illustration

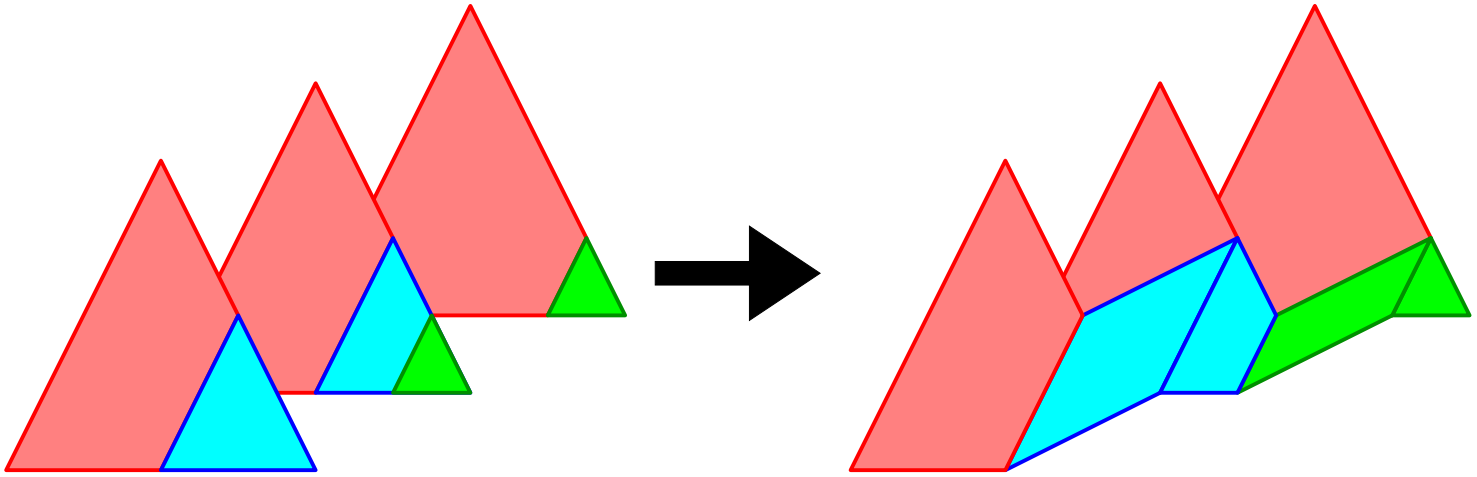




2.3 Component-hypertrees: Simplification

Simplification

- Several nodes / subtrees need to be represented once
  - The “dummy” root $\mathbb{Z}^n$
  - The subtrees shared by successive component-trees
    $\rightarrow$ In particular, each singleton $\{x\}$



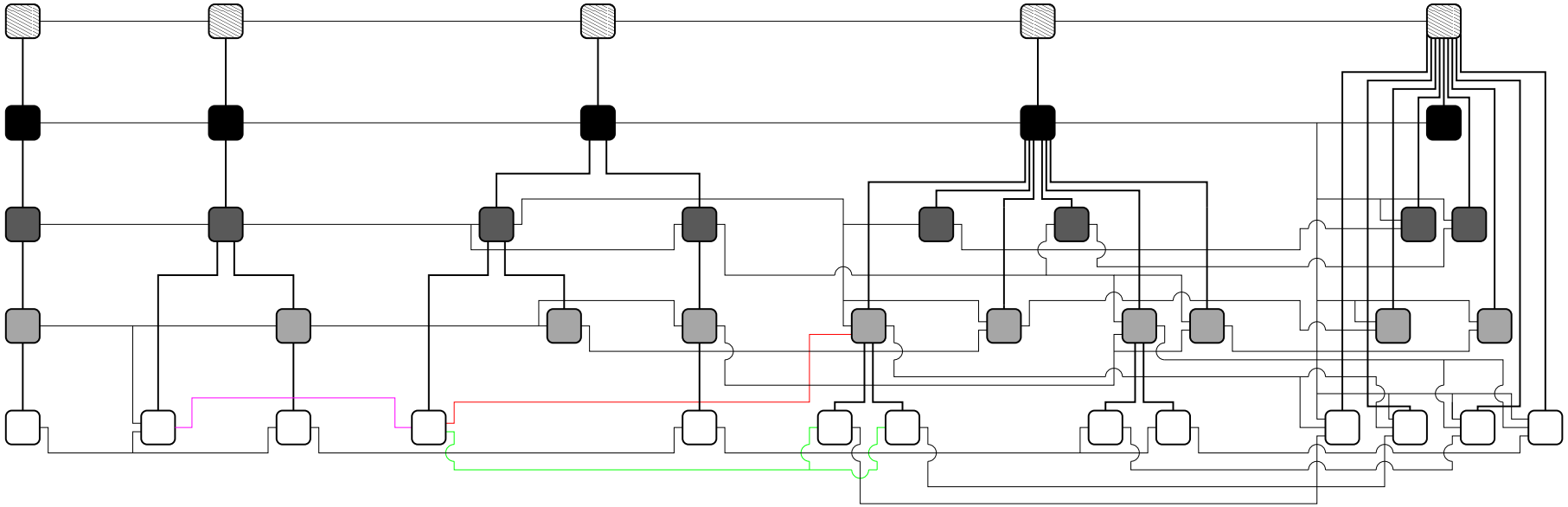
2.3 Component-hypertrees: Simplification

Simplification

- Several nodes / subtrees need to be represented once
  - The “dummy” root $\mathbb{Z}^n$
  - The subtrees shared by successive component-trees
    $\rightarrow$ In particular, each singleton $\{x\}$

$\Rightarrow$ Compact data structure

Before simplification



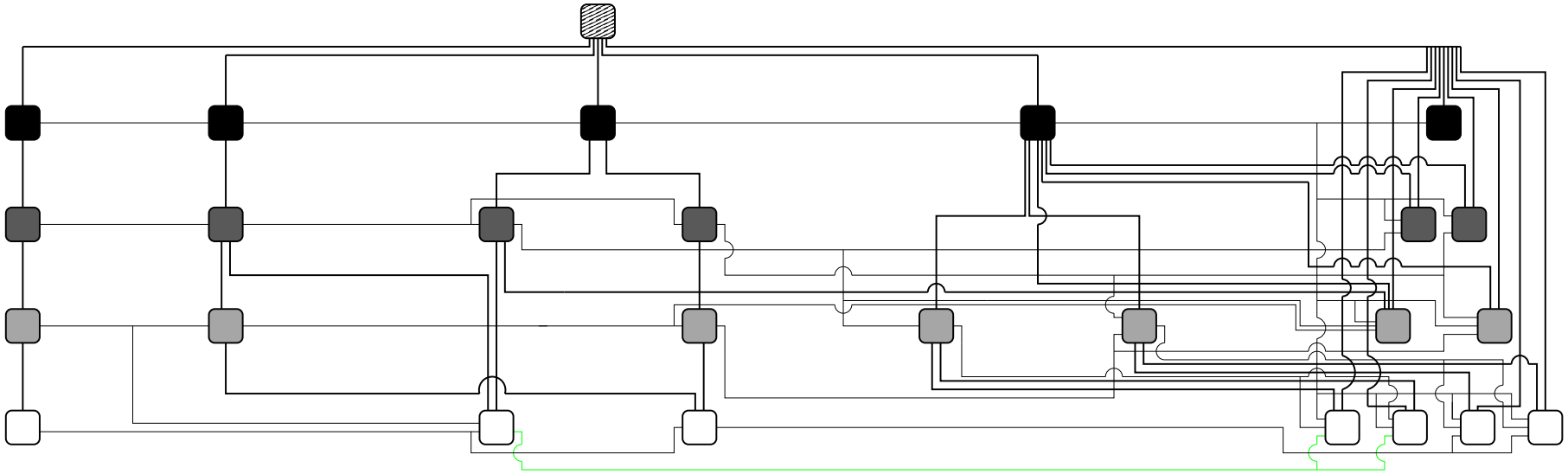
2.3 Component-hypertrees: Simplification

Simplification

- Several nodes / subtrees need to be represented once
  - The “dummy” root $\mathbb{Z}^n$
  - The subtrees shared by successive component-trees
    $\rightarrow$ In particular, each singleton $\{x\}$

$\Rightarrow$ Compact data structure

After simplification



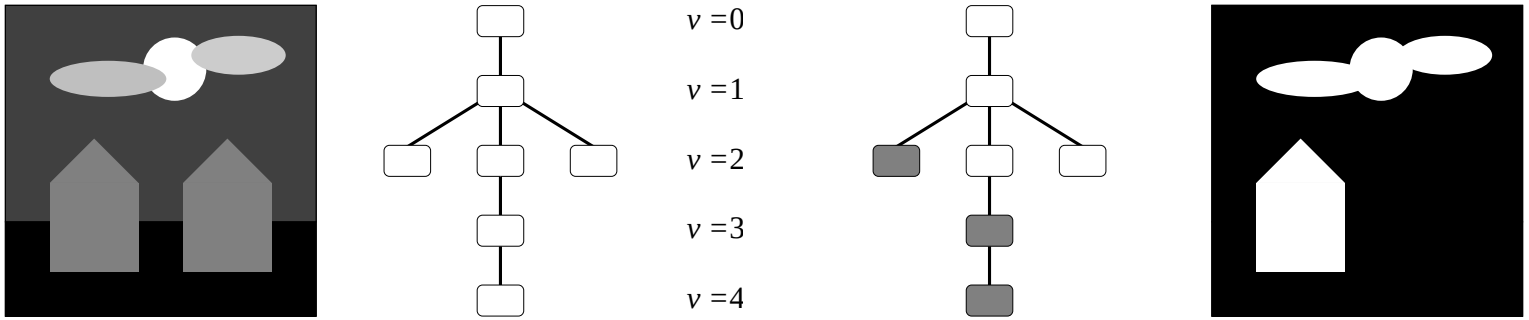
Outline

1. Introduction
2. Component-hypertrees
3. Component-hypertrees and segmentation – A case study
4. Experiments
5. Conclusion

3.1 Previous work: example-based interactive segmentation

Component-tree based segmentation

- Image segmentation $\sim$ Nodes selection
  $\rightarrow$ Generally based on attributes



$I : E \rightarrow V$ | Component-tree T | Subset $\mathcal{K}'$ of nodes selected from T | Binary image $\bigcup_{N \in \mathcal{K}'} N$

3.1 Previous work: example-based interactive segmentation

An example-based segmentation method (Passat, Naegel, 2011)

- Selection of the nodes which match at best *a user-defined presegmentation*
- Image $I : E \to V$; T its component-tree, $G \subseteq E$ a binary marker (presegmentation) $\to$ Optimisation problem

$$\widehat{\mathcal{K}} = \arg \min_{\mathcal{K}' \in 2^{\mathcal{K}}} \left\{ d^{\alpha}\Big(\bigcup_{N \in \mathcal{K}'} N, G \Big) \right\}$$

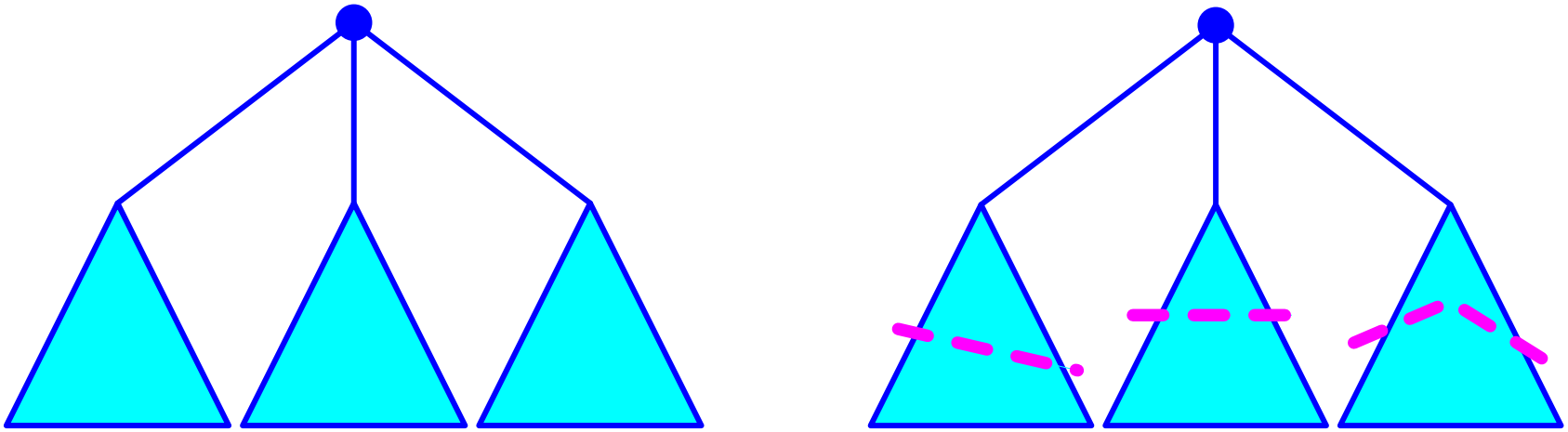
where

- $\mathcal{K}$ is the set of nodes of T
- $d^{\alpha}(X, Y) = \alpha.|X \setminus Y| + (1 - \alpha).|Y \setminus X|$ (false positives vs. false negatives)

3.1 Previous work: example-based interactive segmentation

Example-based approach (contd.)

- The set $G^\alpha = \bigcup_{N' \in \mathcal{K}'} N'$ minimizes $d^\alpha(., G)$, with $\mathcal{K}'$ composed of all the nodes N which are set "on"
  $\rightarrow$ recursive procedure
- Algorithmic complexity $\mathcal{O}(\max\{|\mathcal{K}|, |E|\})$



3.1 Previous work: example-based interactive segmentation

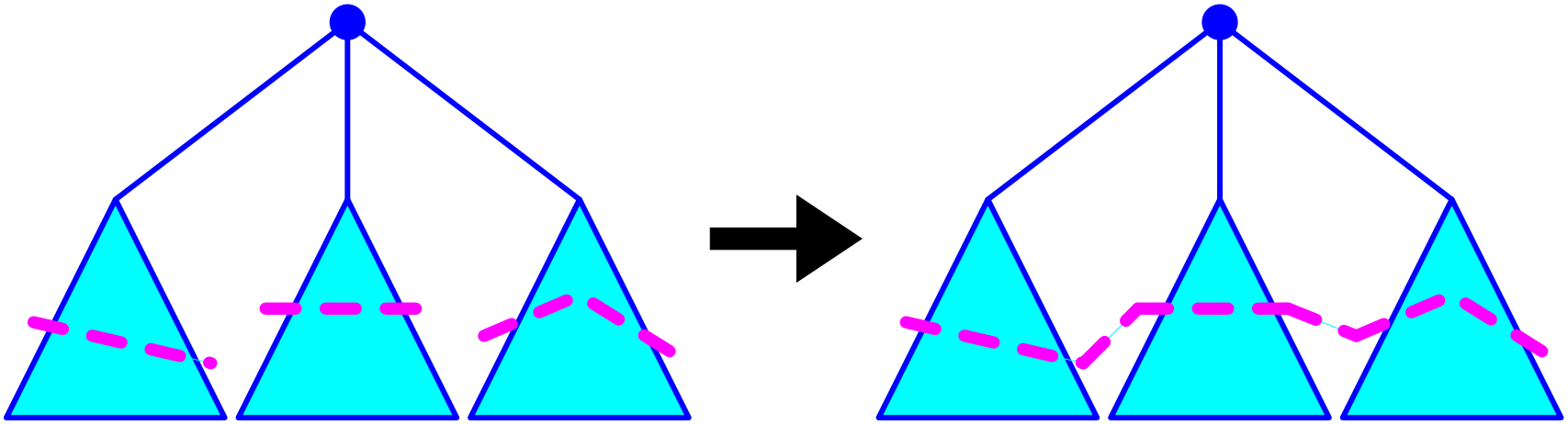
Example-based approach (contd.)

- A node N is "on" and $c^{\alpha}(N) = \alpha.n(N,G)$ if

$$\alpha.n(N,G) \prec (1-\alpha).p^*(N,G) + \sum_{N' \in ch(N)} c^{\alpha}(N')$$

with :

- $ch(N)$ the set of children of N
- $n(N,G) = |N \setminus G|$
- $p^*(N,G) = |(N \setminus \bigcup_{N' \in ch(N)} N') \cap G|$



3.1 Previous work: example-based interactive segmentation

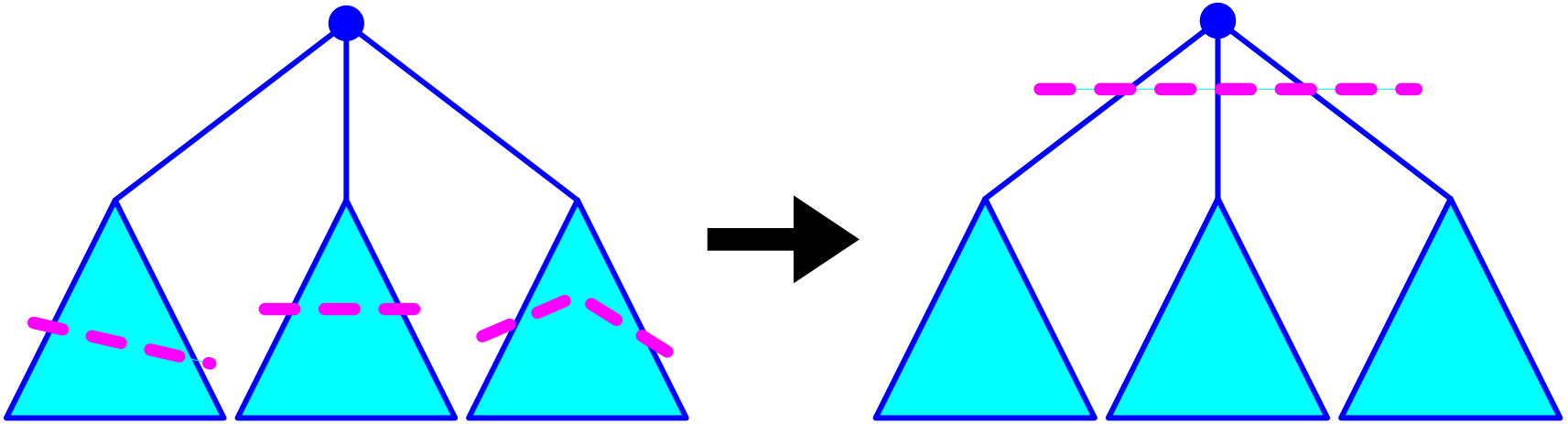
Example-based approach (contd.)

- A node N is "off" and $c^\alpha(N) = (1-\alpha).p^*(N,G) + \sum_{N' \in ch(N)} c^\alpha(N')$ if

$$\alpha.n(N,G) \not< (1-\alpha).p^*(N,G) + \sum_{N' \in ch(N)} c^\alpha(N')$$

with :

- $ch(N)$ the set of children of N
- $n(N,G) = |N \setminus G|$
- $p^*(N,G) = |(N \setminus \bigcup_{N' \in ch(N)} N') \cap G|$

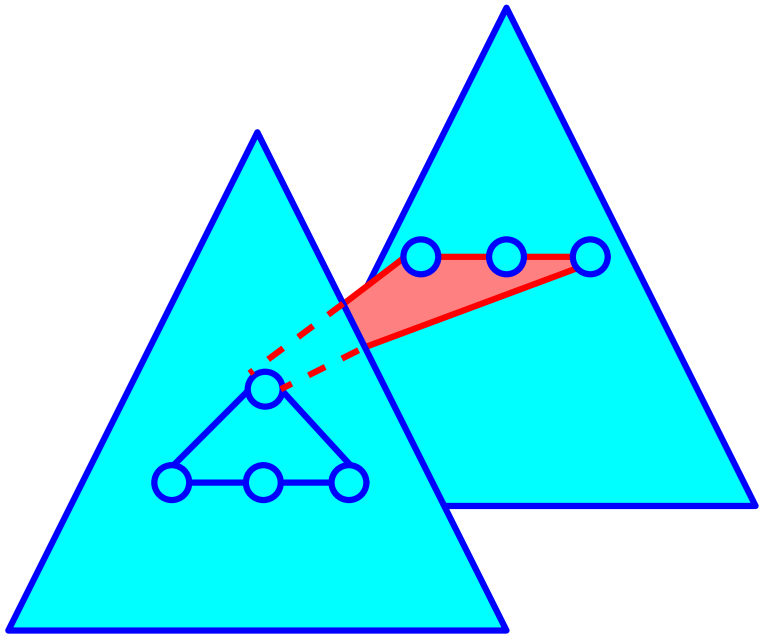


3.2 Extension to component-hypertrees

Extension of the method

Method devoted to component-trees (defined for a given connectivity)
$\rightarrow$ Extension to multiple connectivities / to component-hypertrees ?

- Naive solution: Application to each component-tree of the component-hypertree
- Alternative solution: Modify the algorithmic process to fit component-hypertrees

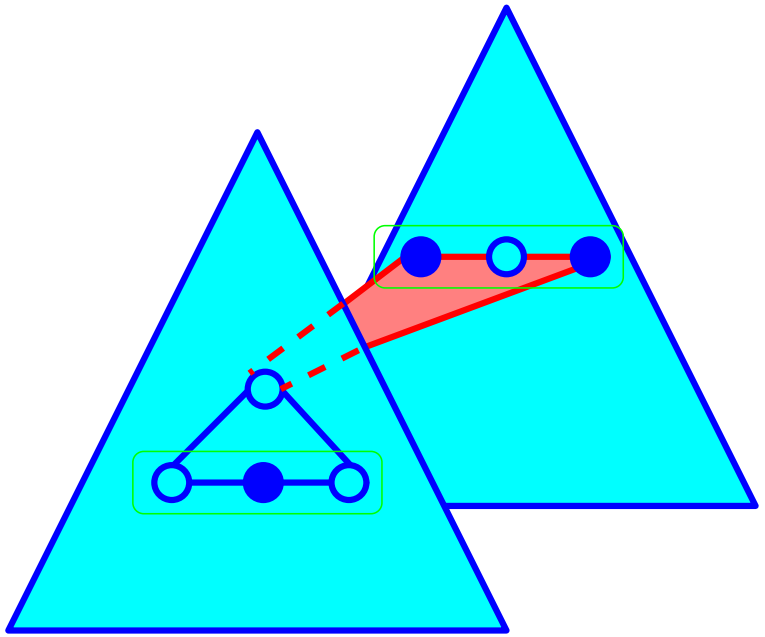


3.2 Extension to component-hypertrees

Example-based approach with component-hypertrees

- Instead of using
  - $n(N, G)$
  - $p^*(N, G)$
  - $c^\alpha(N')$

  use of
  - $\delta^+(N) = \alpha.n(N, G) - (1 - \alpha).p^*(N, G) - \sum_{N' \in ch(N)} c^\alpha(N')$
  - $\delta^-(N) = c^\alpha(N) - \sum_{D \in d(N)} c^\alpha(D)$

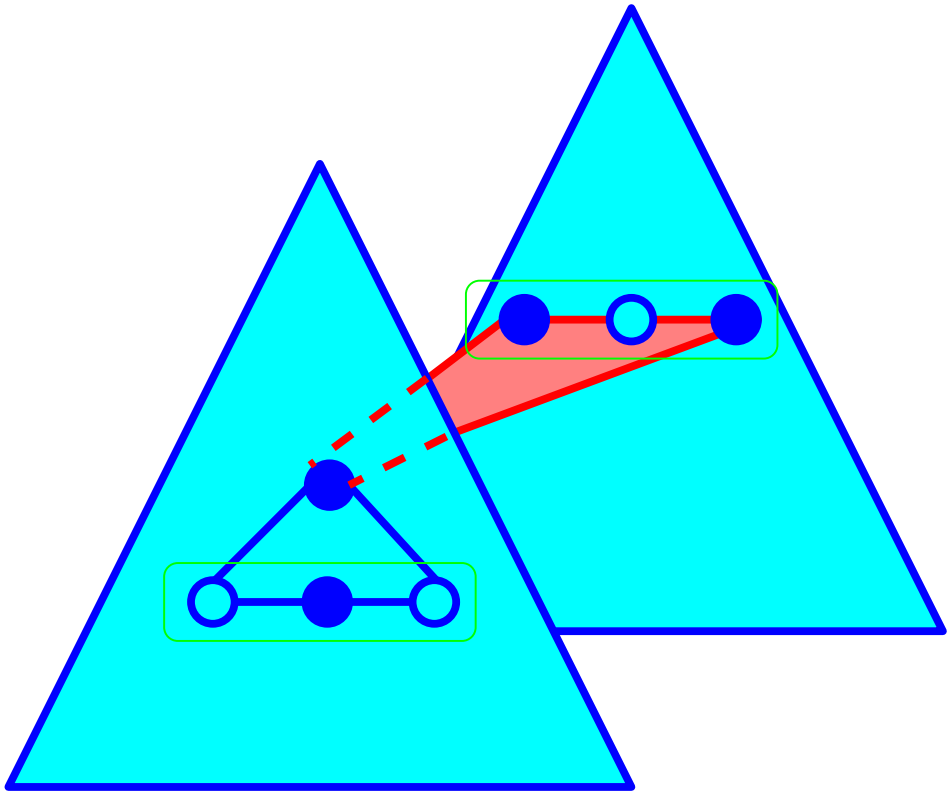


3.2 Extension to component-hypertrees

Example-based approach with component-hypertrees (contd.)

- N is “on” if

$$\sum_{D \in d(N)} \delta^{+}(D) \prec \sum_{N' \in ch(N)} \delta^{-}(N')$$

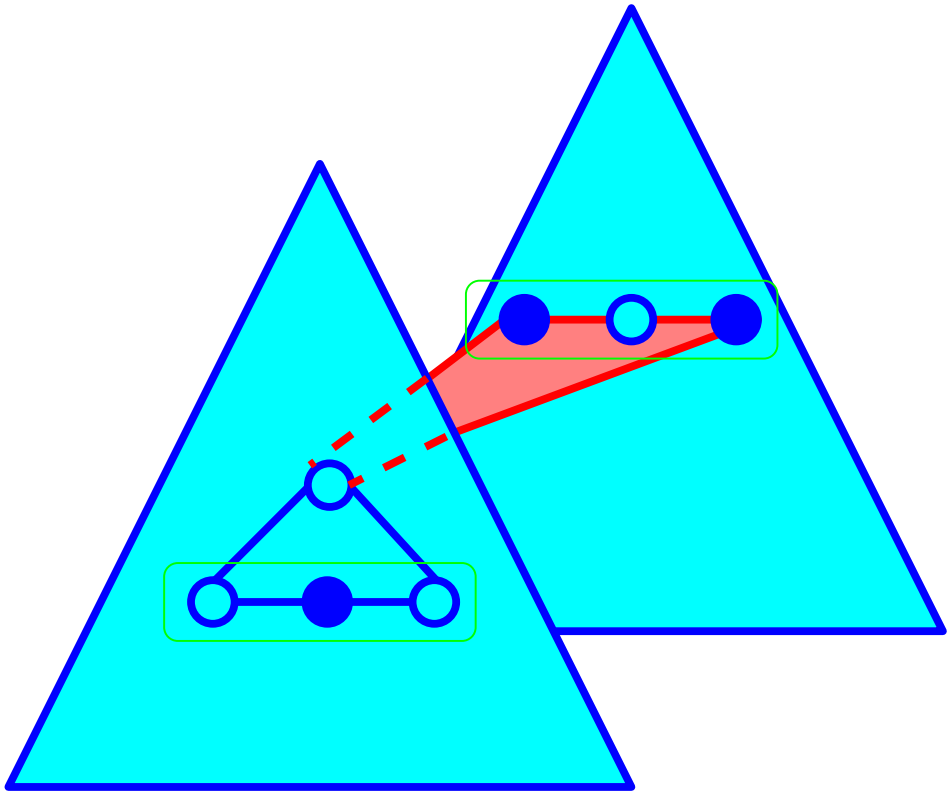


3.2 Extension to component-hypertrees

Example-based approach with component-hypertrees (contd.)

- N is “off” if

$$\sum_{D \in d(N)} \delta^{+}(D) \not< \sum_{N' \in ch(N)} \delta^{-}(N')$$



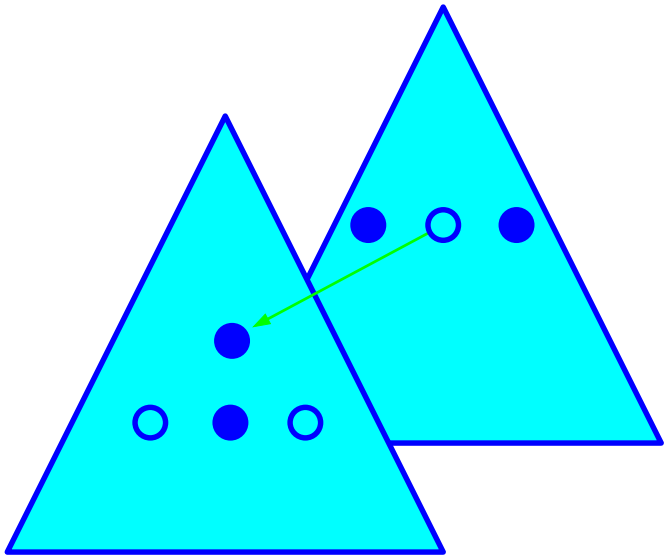
3.2 Extension to component-hypertrees

Algorithmics

- Recursive computation of δ^- and δ^+
- "Bottom-up/right-left" algorithmic process to compute recursively the solution

N is "on": computation of δ^-

- $\delta^-(N) = \sum_{D \in d_\circ(N)} \delta^+(D)$



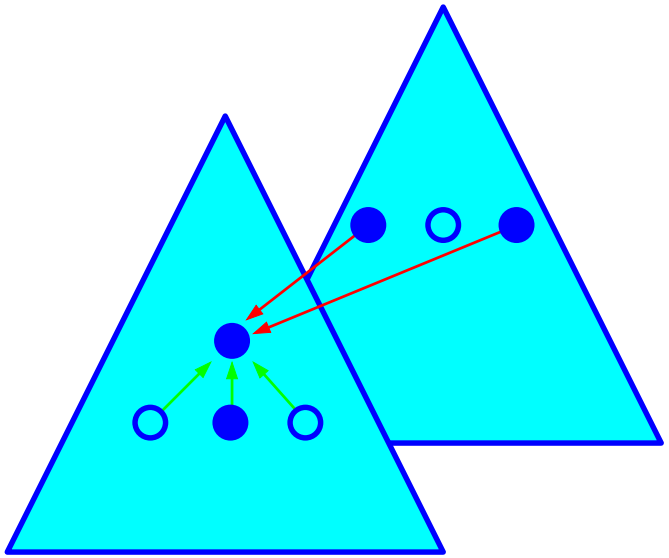
3.2 Extension to component-hypertrees

Algorithmics

- Recursive computation of δ^- and δ^+
- "Bottom-up/right-left" algorithmic process to compute recursively the solution

N is "off": computation of δ^-

- $\delta^-(N) = \sum_{N' \in ch(N)} \delta^-(N') - \sum_{D \in d_\bullet(N)} \delta^+(D)$



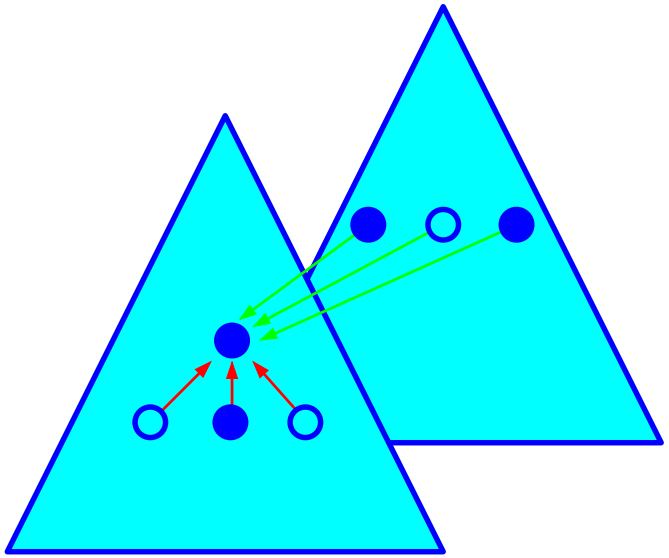
3.2 Extension to component-hypertrees

Algorithmics

- Recursive computation of δ^- and δ^+
- "Bottom-up/right-left" algorithmic process to compute recursively the solution

Computation of δ^+

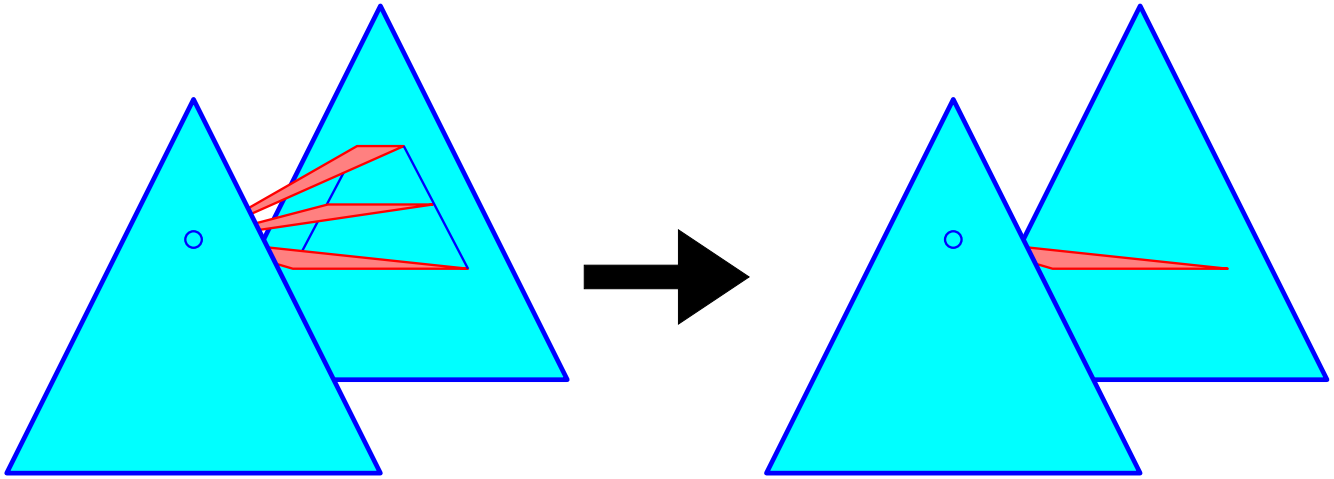
- $\delta^+(N) = \sum_{D \in d(N)} \delta^+(D) - \sum_{N' \in ch(N)} \delta^-(N')$



3.2 Extension to component-hypertrees

Optimisation 1: Data structure simplification

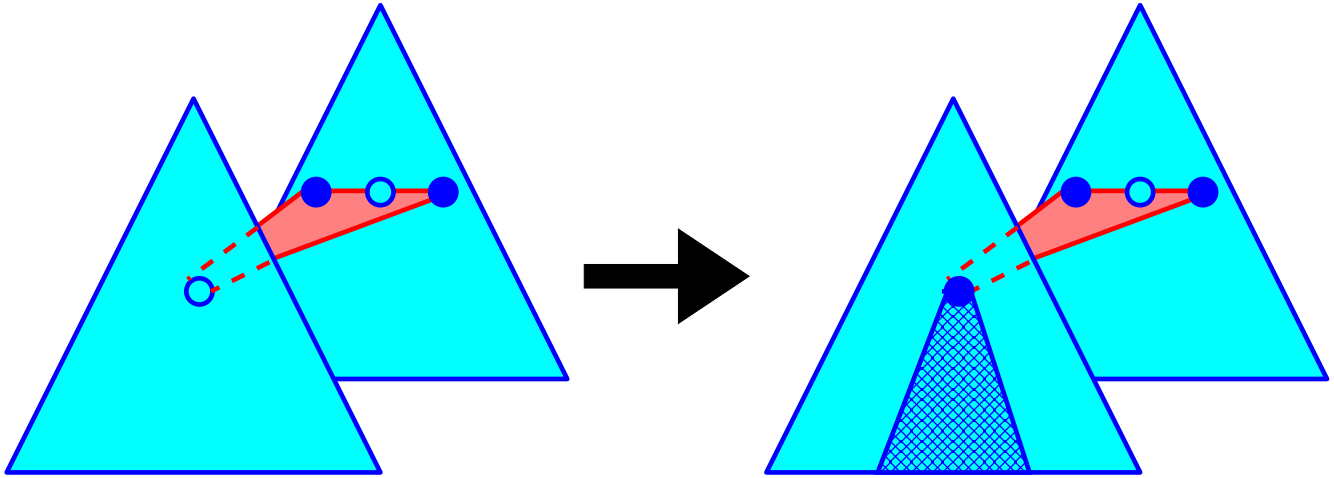
- If a node N has several decompositions in the partition trees
  $\rightarrow$ only preserves the finest one



3.2 Extension to component-hypertrees

Optimisation 2: Minimizing computation

- If $\sum_{D \in d(N)} \delta^{+}(D) \prec 0$ (true if all the nodes are "on" in $d(N)$)
  $\rightarrow$ do not process the subtrees of N



Outline

1. Introduction
2. Component-hypertrees
3. Component-hypertrees and segmentation – A case study
4. Experiments
5. Conclusion

4. Experimentations

Anti-extensive mask

I and marker G (in red)

Result using standard connectivity

$\rightarrow$ undersegmentation

4. Experimentations

Anti-extensive mask

I and marker G (in red)

Result using mask-based connectivity : $\Omega(I) = \gamma_1(I)$

4. Experimentations

Extensive mask

I and marker G (in red)

Result using standard connectivity

4. Experimentations

Extensive mask

I and marker G (in red)

Result using mask-based connectivity : $\Omega(I) = \varphi_2(I)$

Outline

1. Introduction
2. Component-hypertrees
3. Component-hypertrees and segmentation – A case study
4. Experiments
5. Conclusion

5. Conclusion

Main contributions

- Extension of the component-trees to the case of multiple connectivities: component-hypertree
- Example-based segmentation method for component-hypertrees
- Application to dropcaps segmentation

Perspectives

- Use of increasingness properties to optimize the process for several α values
- In the case where V is a partially-ordered set: extension to component-graphs