1D Component Tree in Linear Time and Space

and its Application to Gray-Level Image Multithresholding

David Menotti[1 2] Laurent Najman[1]
Arnaldo de Albuquerque Araújo[2]

[1]Institut Gaspard-Monge - Université Paris-Est - LA^{2}SI, Groupe ESIEE-Paris, France
{d.menotti,l.najman}@esiee.fr

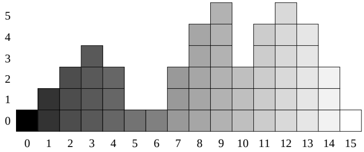
[2]Universidade Federal de Minas Gerais - NPDI, DCC, Belo Horizonte, MG, Brazil
{menotti,arnaldo}@dcc.ufmg.br

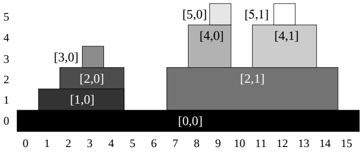
ISMM'2007 - 10-13 October 2007

Outline

What Is the Component Tree?

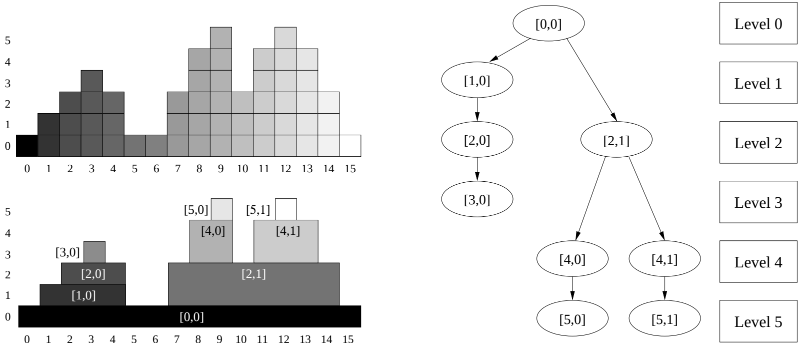
- Component/Level > Inclusion Relation > Tree Structure





What Is the Component Tree?

- Component/Level > Inclusion Relation > Tree Structure



Applications

- The component tree captures essential features of a signal
  - The attributes: the volume, surface and height
- Applications
  - Image Filtering and Segmentation [Jones, 1997, Najman & Couprie, 2006]
  - Video Segmentation [Salembier et al., 1998]
  - Image Registration [Mattes et al., 1999]
  - Image Compression [Salembier et al., 1998]

Other Algorithms

- Morphological Operations - [Breen & Jones, 1996]
- Study of times complexity - [Mattes & Demongeot, 2000]
- $O(n \times L)$ [Salembier et al., 1998]
  - The fastest one for practical use
- A quasi linear $O(n \times \alpha(n))$ - [Najman & Couprie, 2006]
  - where $\alpha(10^{80}) \approx 4$
- We propose a **linear** time and space algorithm to compute the component tree for **1D signals**

Ordered Set

- Let P be a ***set of points***
- Let $\prec$ be a ***binary relation*** on P ($\prec \subseteq P \times P$), which is
  - **transitive** ($(x, y) \in \prec, (y, z) \in \prec \Rightarrow (x, z) \in \prec$), and
  - **trichotomous** (*i.e.*, exactly one of $(x, y) \in \prec, (y, x) \in \prec$ and $x = y$ is true)

Ordered Set

- Let P be a ***set of points***
- Let $\prec$ be a ***binary relation*** on P ($\prec \subseteq P \times P$), which is
  - **transitive** ($(x, y) \in \prec, (y, z) \in \prec \Rightarrow (x, z) \in \prec$), and
  - **trichotomous** (*i.e.*, exactly one of $(x, y) \in \prec, (y, x) \in \prec$ and $x = y$ is true)
- $(P, \prec)$ - (totally) ***ordered set***

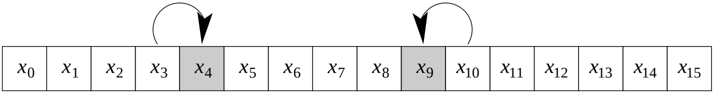
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

Predecessor and Successor on Ordered Set

- Let $(P, \prec)$ be an ordered set
- If $(x, y) \in \prec$ and there is no z such that $(x, z) \in \prec$ and $(z, y) \in \prec$
  - y is the ***successor*** of x
  - x is the ***predecessor*** of y

Predecessor and Successor on Ordered Set

- Let $(P, \prec)$ be an ordered set
- If $(x, y) \in \prec$ and there is no z such that $(x, z) \in \prec$ and $(z, y) \in \prec$
  - y is the ***successor*** of x
  - x is the ***predecessor*** of y



- x_4 is the successor of x_3
- x_9 is the predecessor of x_{10}

Connected Set on Ordered Set

- Let $(P, <)$ be an ordered set
- Let $X = \{x_0, x_1, ..., x_n\} \subseteq P$
  where $x_0, x_1, ..., x_n$ are arranged in increasing order $((P, <))$
- If for any $i \in [1, n]$, x_i is the successor of x_{i-1}, then we say that X is a ***connected set***

Connected Set on Ordered Set

- Let $(P, \prec)$ be an ordered set
- Let $X = \{x_0, x_1, ..., x_n\} \subseteq P$
  where $x_0, x_1, ..., x_n$ are arranged in increasing order $((P, \prec))$
- If for any $i \in [1, n]$, x_i is the successor of x_{i-1}, then we say that X is a ***connected set***

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

Connected Set on Ordered Set

- Let $(P, \prec)$ be an ordered set
- Let $X = \{x_0, x_1, ..., x_n\} \subseteq P$
  where $x_0, x_1, ..., x_n$ are arranged in increasing order $((P, \prec))$
- If for any $i \in [1, n]$, x_i is the successor of x_{i-1}, then we say that X is a ***connected set***

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

x_3	x_4	x_5

x_8	x_9	x_{10}	x_{11}	x_{12}

The Starting and Ending Points

- Let $(P, \prec)$ be an ordered set
- Let $X = \{x_3, x_4, ..., x_{10}\} \subseteq P$ be a *connected set*

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}

The Starting and Ending Points

- Let $(P, \prec)$ be an ordered set
- Let $X = \{x_3, x_4, ..., x_{10}\} \subseteq P$ be a *connected set*

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}

- x_3 is the ***starting point*** of X
- x_{10} is the ***ending point*** of X

The Weighted Ordered Set

- Let $(P, \prec)$ be an ordered set
- Let $\mathcal{F}(P, D)$ be the set composed of all **mappings** from P to D (*e.g.*, $D \subseteq \mathbb{N}$)
- For a $F \in \mathcal{F}$, $(P, \prec, F)$ is called a ***weighted ordered set*** (WOS)
- For a point $p \in P$, $F(p)$ is called the ***weight*** (or ***level***) of p

The Weighted Ordered Set

- Let $(P, \prec)$ be an ordered set
- Let $\mathcal{F}(P, D)$ be the set composed of all **mappings** from P to D (*e.g.*, $D \subseteq \mathbb{N}$)
- For a $F \in \mathcal{F}$, $(P, \prec, F)$ is called a ***weighted ordered set*** (WOS)
- For a point $p \in P$, $F(p)$ is called the ***weight*** (or ***level***) of p

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
0	1	2	3	2	0	0	2	4	5	2	4	5	4	2	0

- $F(x_7) = 2$ is the weight/level of x_7

The Upper-Weighted Set / The Connected Component

- Upper-weighted set $F_h = \{p \in P | F(p) \geq h\}$
- A connected set X of an upper-weighted set which is ***maximal*** (*i.e.*, $X = Y$ whenever $X \subseteq Y \subseteq P$ and Y is connected) is called a ***connected component***

F	0	1	2	3	2	0	0	2	4	5	2	4	5	4	2	0
F_0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F_1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_2	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_3	0	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0
F_4	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0
F_5	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

The Upper-Weighted Set / The Connected Component

- Upper-weighted set $F_h = \{p \in P | F(p) \geq h\}$
- A connected set X of an upper-weighted set which is ***maximal*** (*i.e.*, $X = Y$ whenever $X \subseteq Y \subseteq P$ and Y is connected) is called a ***connected component***

F	0	1	2	3	2	0	0	2	4	5	2	4	5	4	2	0
F_0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F_1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_2	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_3	0	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0
F_4	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0
F_5	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

The Upper-Weighted Set / The Connected Component

- Upper-weighted set $F_h = \{p \in P | F(p) \geq h\}$
- A connected set X of an upper-weighted set which is ***maximal*** (*i.e.*, $X = Y$ whenever $X \subseteq Y \subseteq P$ and Y is connected) is called a ***connected component***

F	0	1	2	3	2	0	0	2	4	5	2	4	5	4	2	0
F_0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F_1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_2	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_3	0	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0
F_4	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0
F_5	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

The Proper Component

- Let $(P, <, F)$ be a WOS, and $s \subseteq P$ a connected component
- $f(s) = max\{h|s$ is a (h-weighted) connected component of $F\}$
- Let $h = f(s)$, we say that s is a (h***-weighted***) ***proper component of*** F

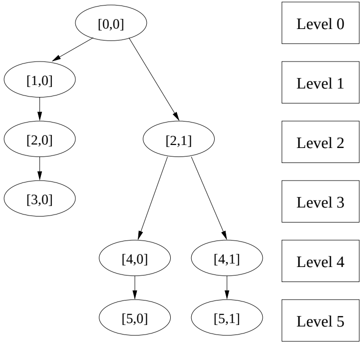
F	0	1	2	3	2	0	0	2	4	5	2	4	5	4	2	0
F_0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F_1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_2	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_3	0	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0
F_4	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0
F_5	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

The Component Tree

- Let $(P, \prec, F)$ be a WOS
- Let $C(F)$ be the set of all components of F
- Let x and y be distinct elements of $C(F)$
  - x is the ***parent*** of y and y is the ***child*** of x, if $y \subset x$ and there is no other $z \in C(F)$ such that $y \subset z \subset x$
- This parent-children relationship, $C(F)$ forms a directed tree named ***component tree*** of F
- Any element/component of $C(F)$ is called a ***node***
- The node that has no parent, is called the ***root*** of the component tree

The Component Tree

F	0	1	2	3	2	0	0	2	4	5	2	4	5	4	2	0
F_0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F_1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_2	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_3	0	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0
F_4	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0
F_5	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

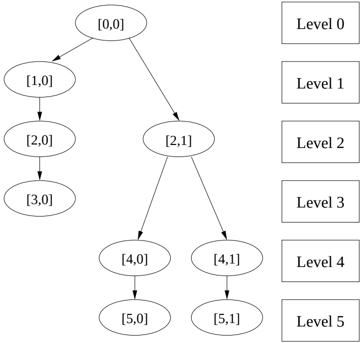


Algorithm Description and Applications

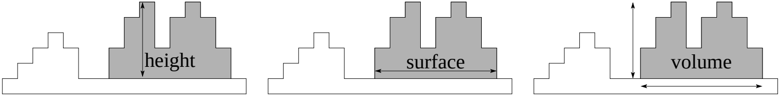
- For sake of algorithm description
  - $c_{h,n} = [h, n]$ - the $(n + 1)$-th h-weighted component of $C(F)$
- Component Mapping - Application
  - Link between the WOS (Signal) and the Component Tree
  - $M(p) = [h, n]$, where $h = F(p)$ and $p \in c_{h,n}$

An Example

F	0	1	2	3	2	0	0	2	4	5	2	4	5	4	2	0
F_0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F_1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_2	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	0
F_3	0	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0
F_4	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0
F_5	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
M h	0	1	2	3	2	0	0	2	4	5	2	4	5	4	2	0
M n	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0



Attributes



Height Surface Volume

$$
\begin{aligned}
ht(c_{h,n}) &= \max_{x \in c_{h,n}} \{F(x) - h_p\}, \\
s(c_{h,n}) &= \text{cardinality}(c_{h,n}), \\
v(c_{h,n}) &= \sum_{x \in c_{h,n}} (F(x) - h_p)
\end{aligned}
$$

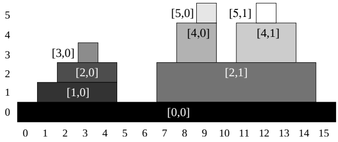
where h_p is the weight/level of the parent of $c_{h,n}$

Our Algorithm

- We propose a linear time and space algorithm to compute the component tree for 1D signals
- No pre-processing is required
  - For instance, extracting local maxima of the signal as done in [Salembier et al., 1998]
- Use of a stack to maintain the relation inclusion

Our Algorithm

- Why is it linear?
  - In an 1D space, components can be determined by their limits (the starting and ending points)
  - The starting and ending points of all components can be detected by processing the signal with a single scan



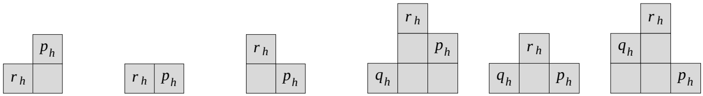
- We do not need to know the exact position of the components in the WOS
- We need to know the components hierarchy (inclusion relation)

Our Algorithm Roughly Works as

- To Build the Component Tree and Component Mapping
- For each point in the WOS, one checks its *status* regarding to its successor
- If a component indicated by a point is found to have descendants it is stored into a stack
- The stack plays a fundamental role to maintain the hierarchy of the component tree
- The parent-children relationships are created as edges between parent and child components

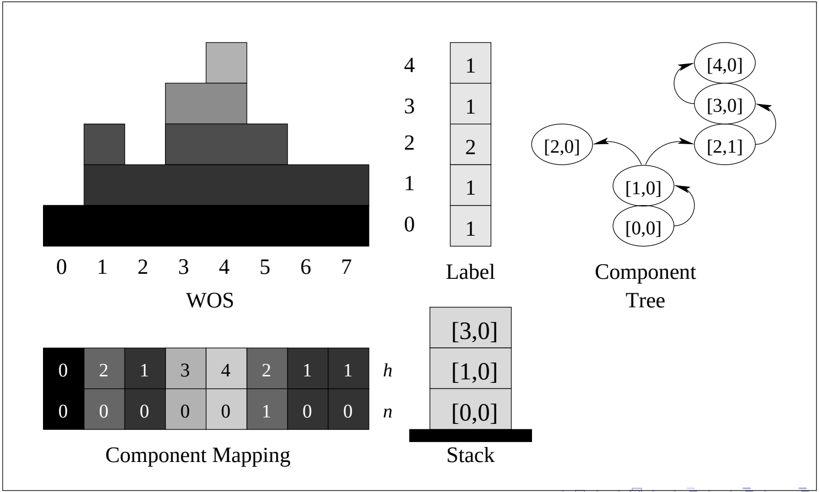
Summary of the Algorithm

- Initialization - The first point of the WOS
- Processing the $n - 1$ points of the WOS
  - The point p ($[p_h, p_n]$) is analyzed based on
  - The point r ($[p_h, p_n]$), which is the predecessor of p, and
  - The point q ($[p_h, p_n]$), which is the stack head

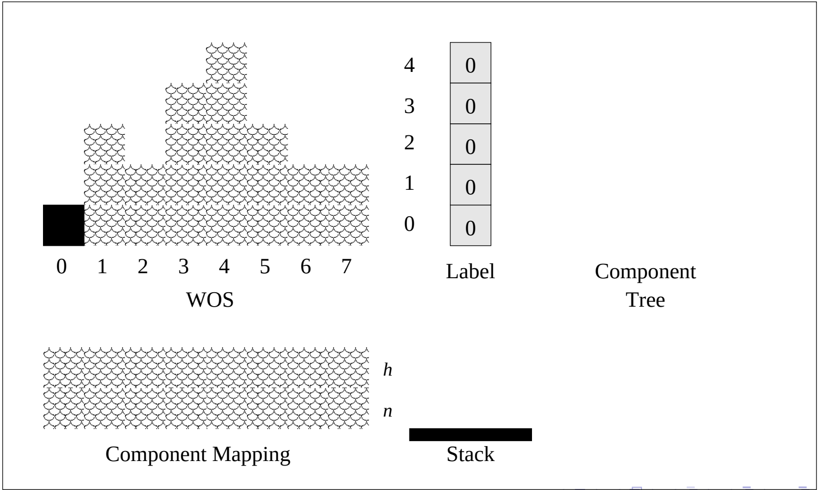


- Finishing - Until the Stack Is Empty

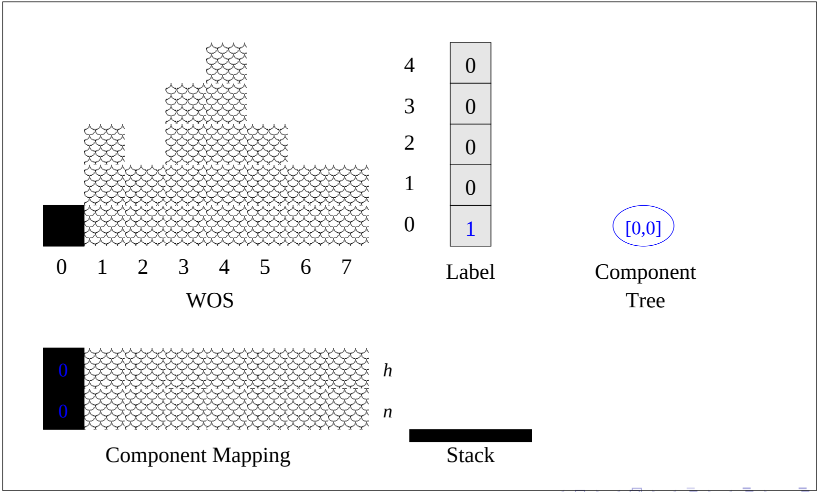
The Structures



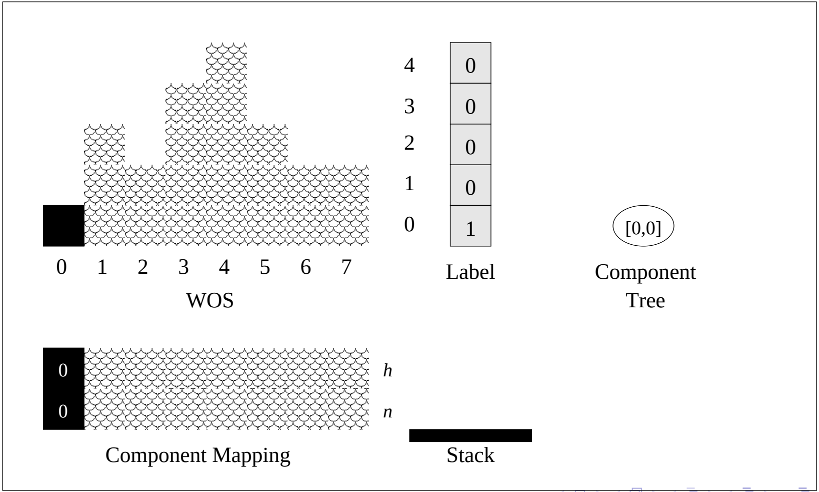
x_0 - Starting Point



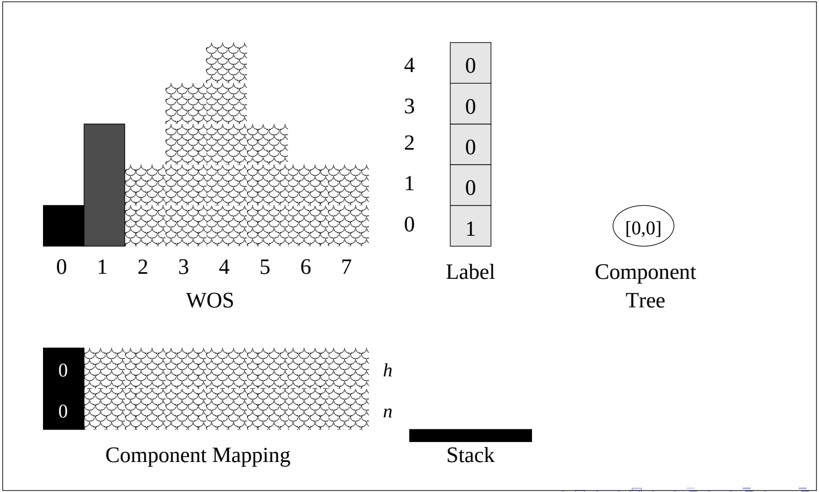
x_0 - Starting Point



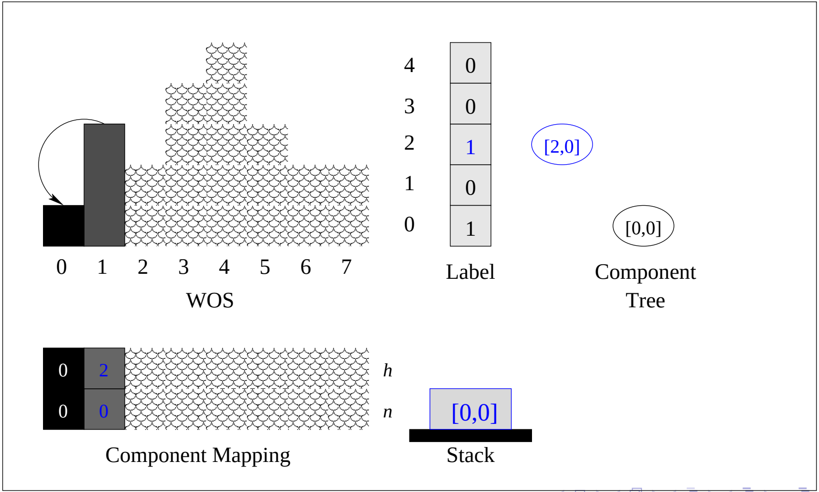
x_0 - Starting Point



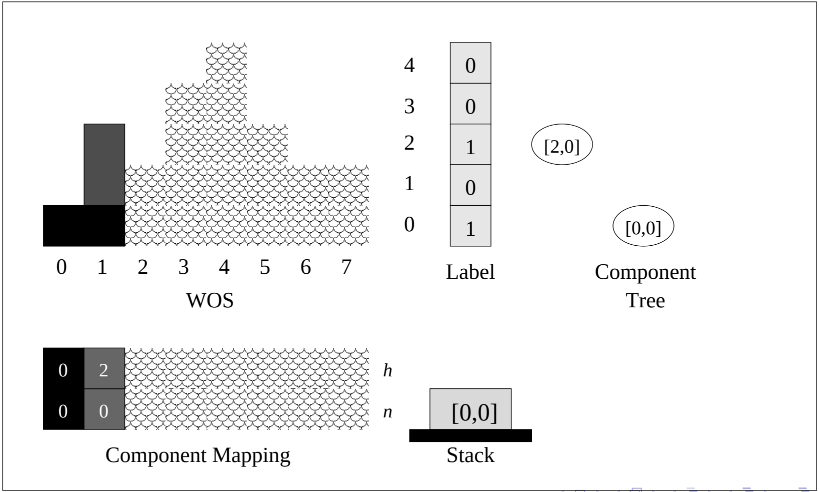
x_1 - $p_h > r_h$ (New Component)



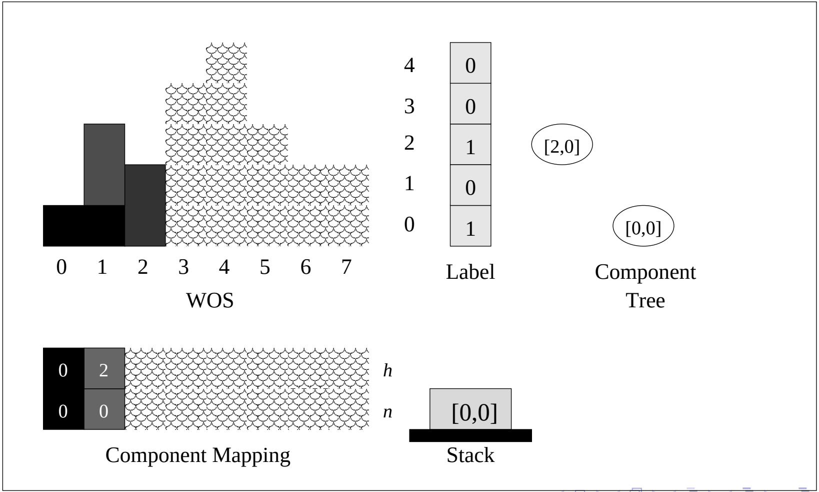
x_1 - $p_h > r_h$ (New Component)



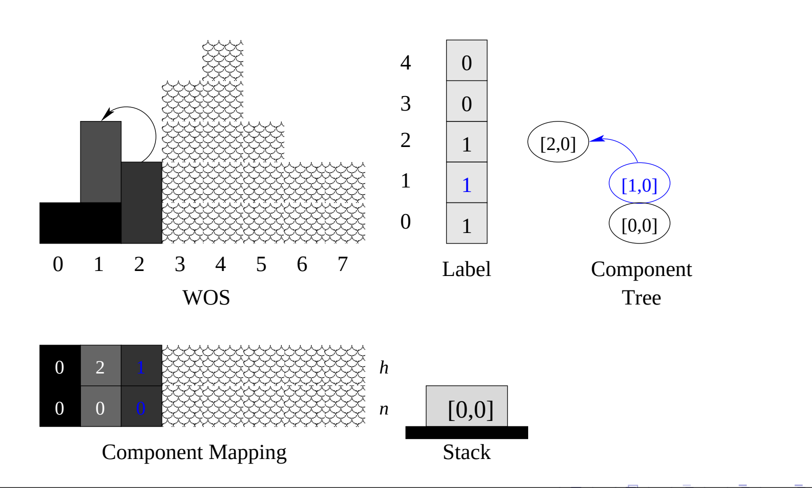
$x_1 - p_h > r_h$ (New Component)



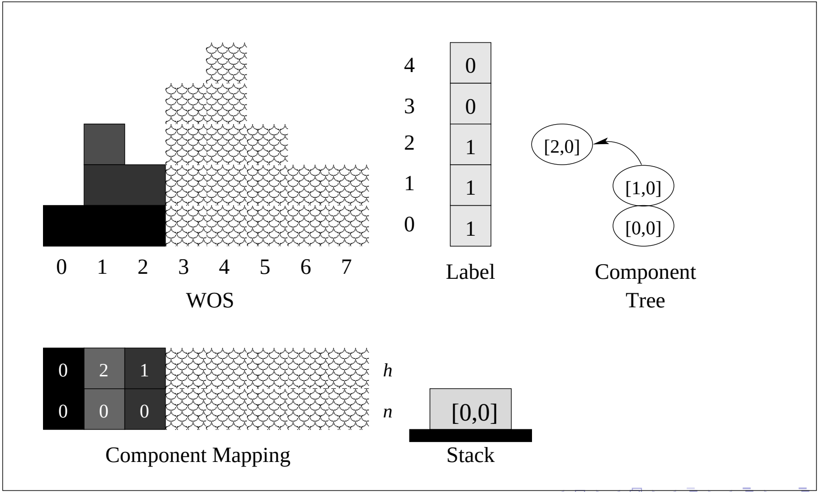
x_2 - $p_h < r_h$ (New Component) and $p_h > q_h$ (Stack)



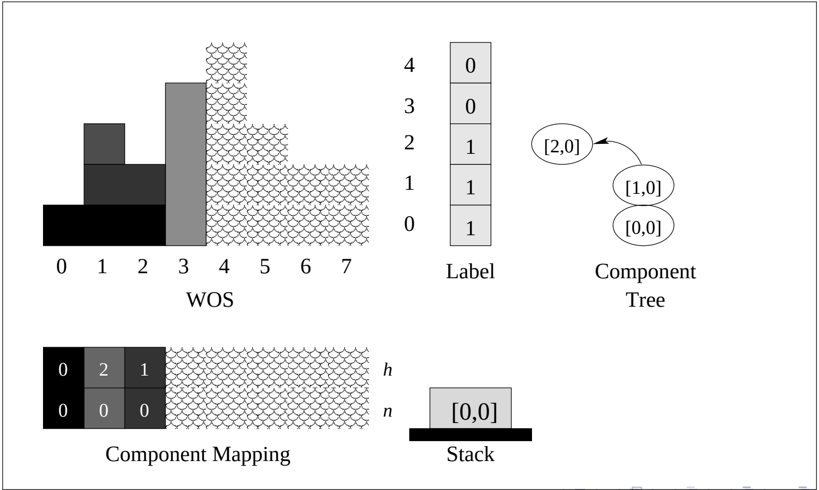
x_2 - $p_h < r_h$ (New Component) and $p_h > q_h$ (Stack)



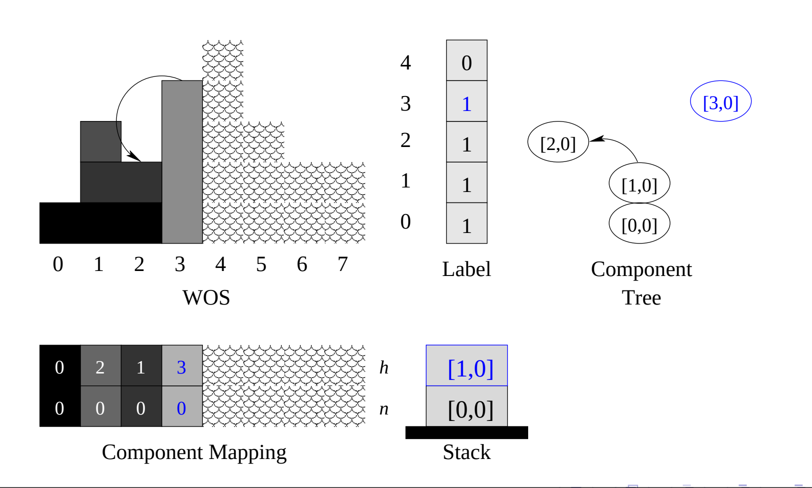
x_2 - $p_h < r_h$ (New Component) and $p_h > q_h$ (Stack)



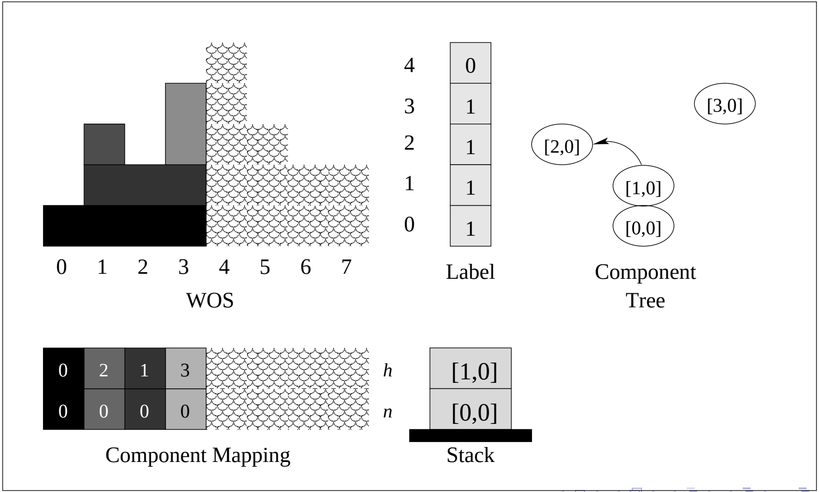
x_3 - $p_h > r_h$ (New Component)



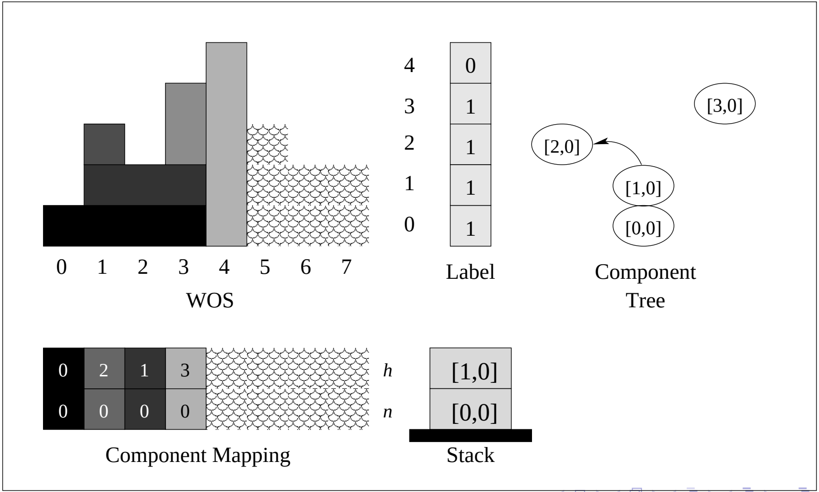
x_3 - $p_h > r_h$ (New Component)



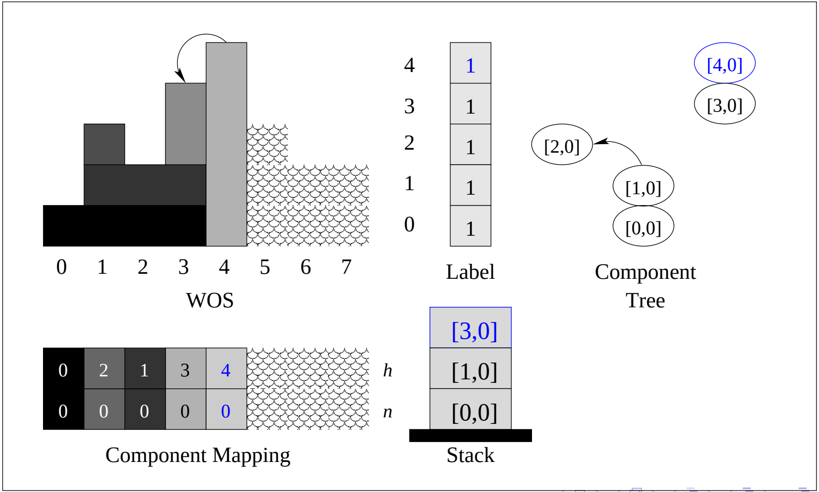
x_3 - $p_h > r_h$ (New Component)



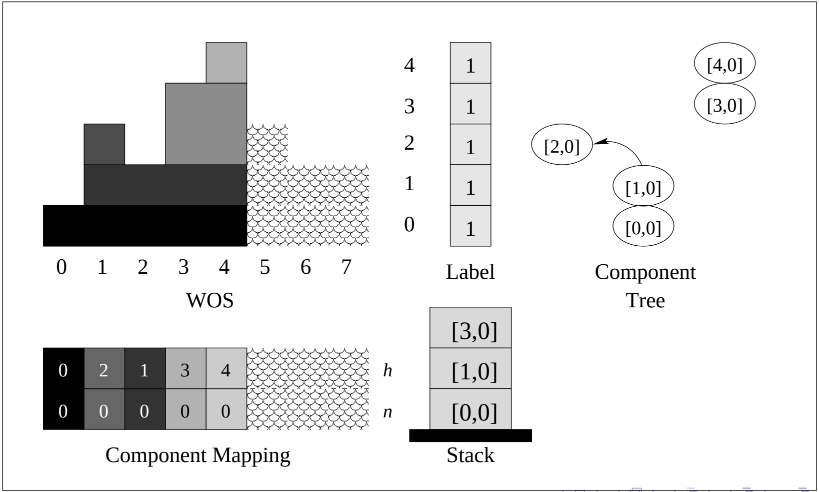
x_4 - $p_h > r_h$ (New Component)



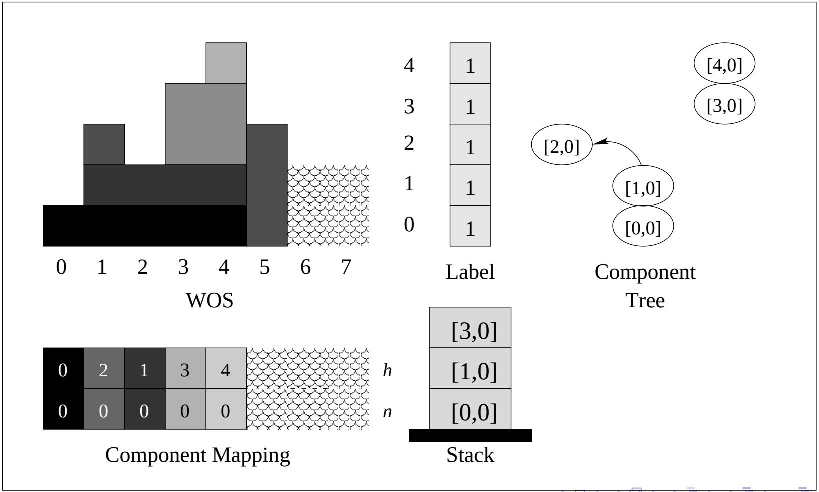
$x_4 - p_h > r_h$ (New Component)



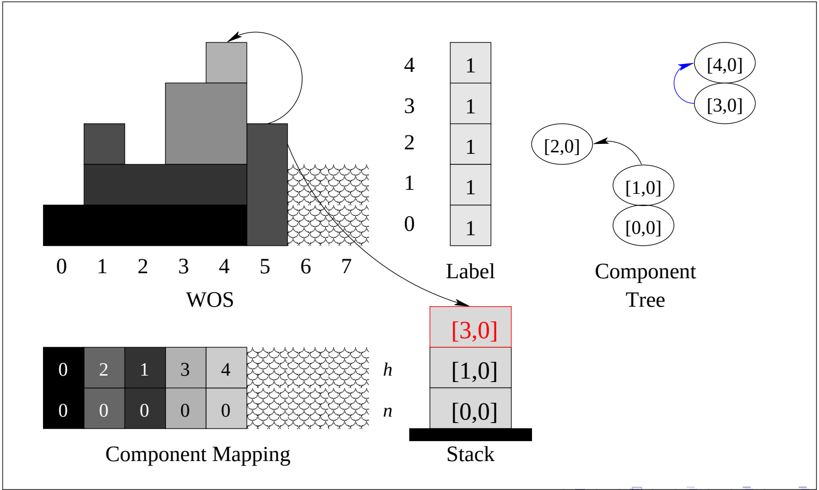
$x_4 - p_h > r_h$ (New Component)



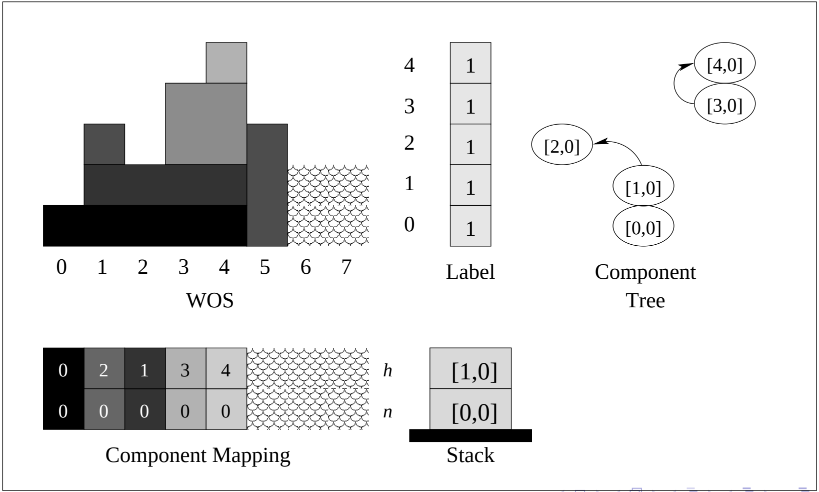
x_5 - $p_h < r_h$ (Ending Point) and $p_h < q_h$ (Stack)



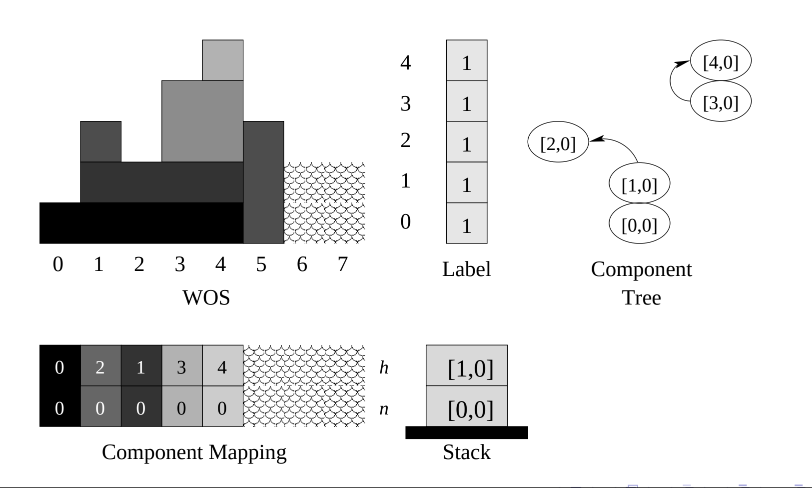
x_5 - $p_h < r_h$ (Ending Point) and $p_h < q_h$ (Stack)



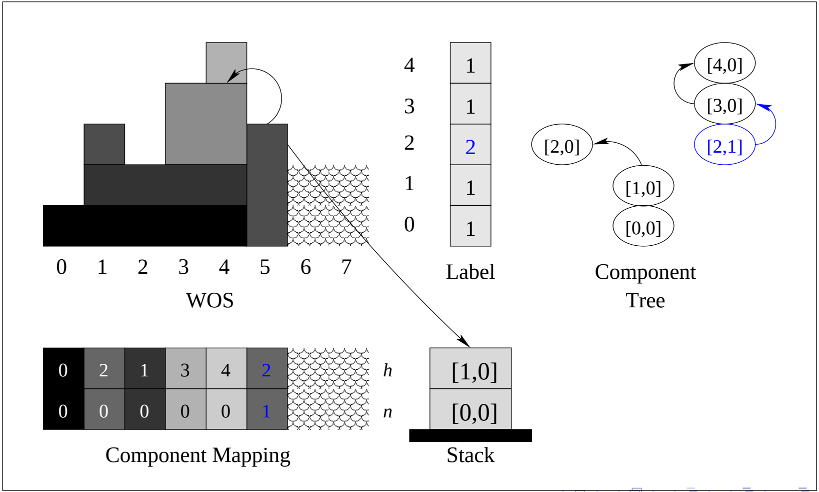
x_5 - $p_h < r_h$ (Ending Point) and $p_h < q_h$ (Stack)



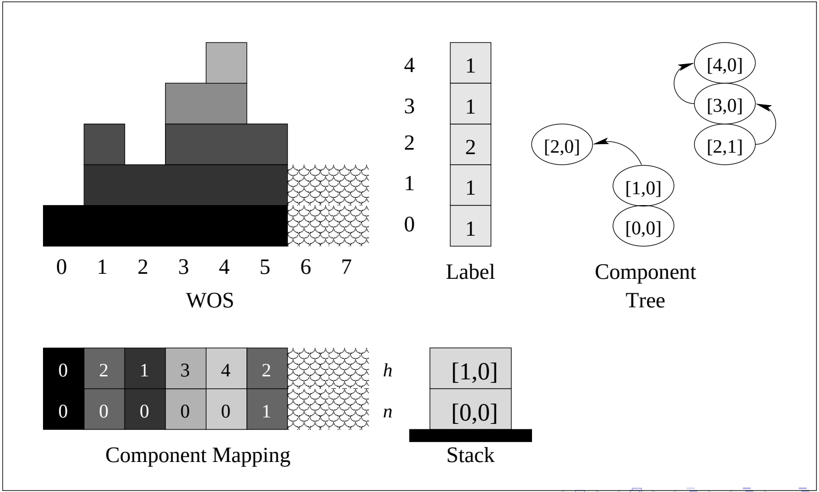
x_5 - $p_h < r_h$ (Ending Point) and $p_h > q_h$ (Stack)



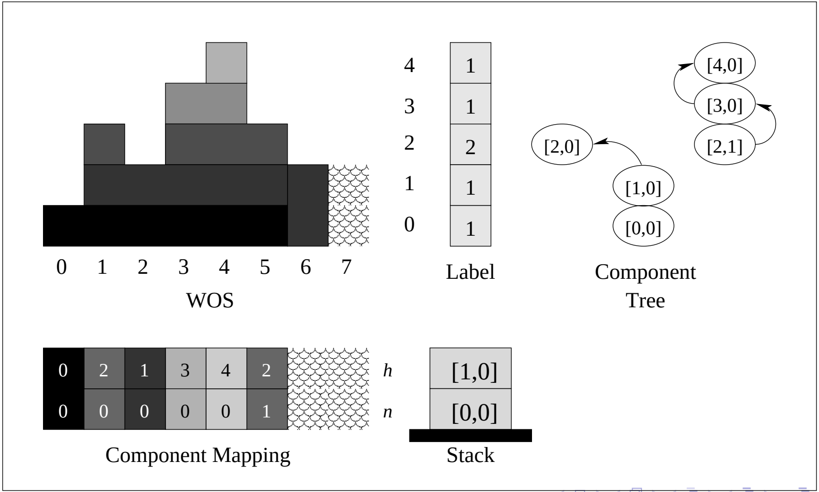
x_5 - $p_h < r_h$ (Ending Point) and $p_h > q_h$ (Stack)



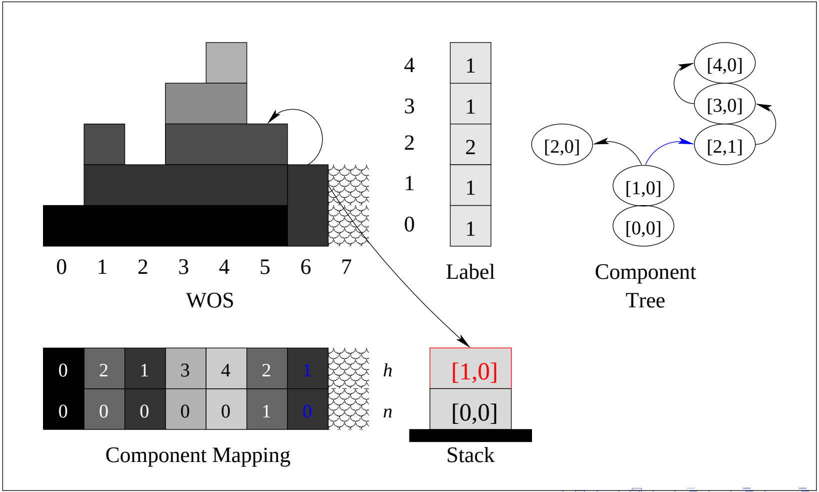
x_5 - $p_h < r_h$ (Ending Point) and $p_h > q_h$ (Stack)



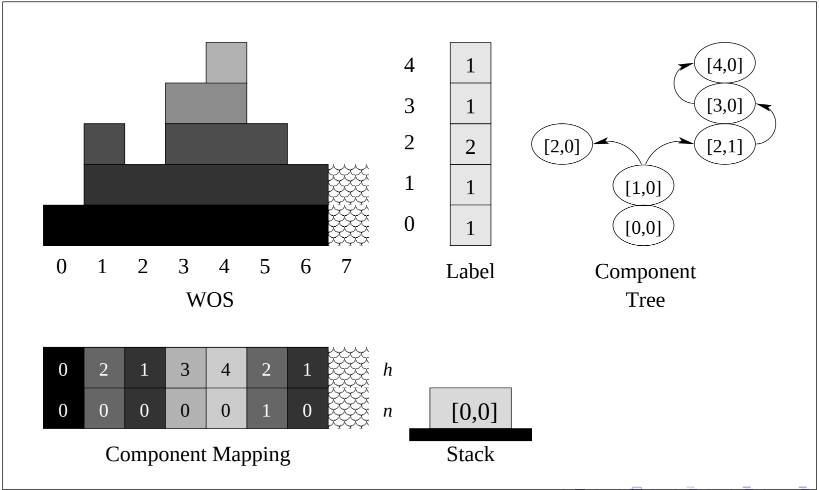
x_6 - $p_h < r_h$ (Ending Point) and $p_h = q_h$ (Stack)



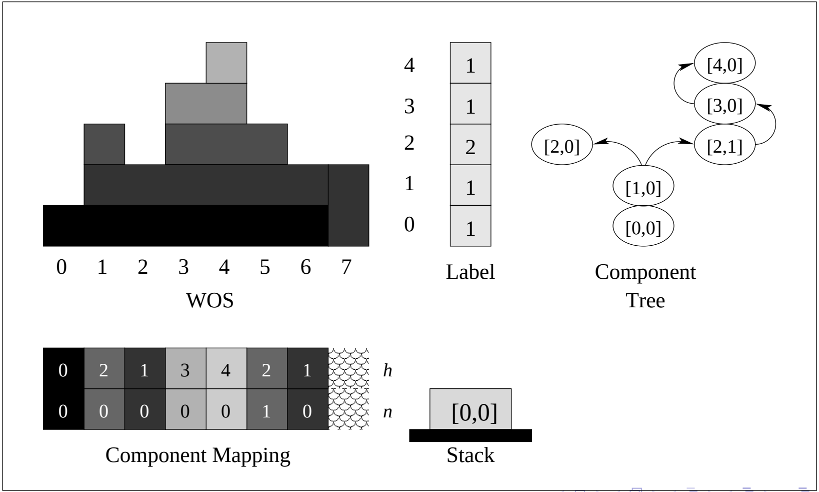
x_6 - $p_h < r_h$ (Ending Point) and $p_h = q_h$ (Stack)



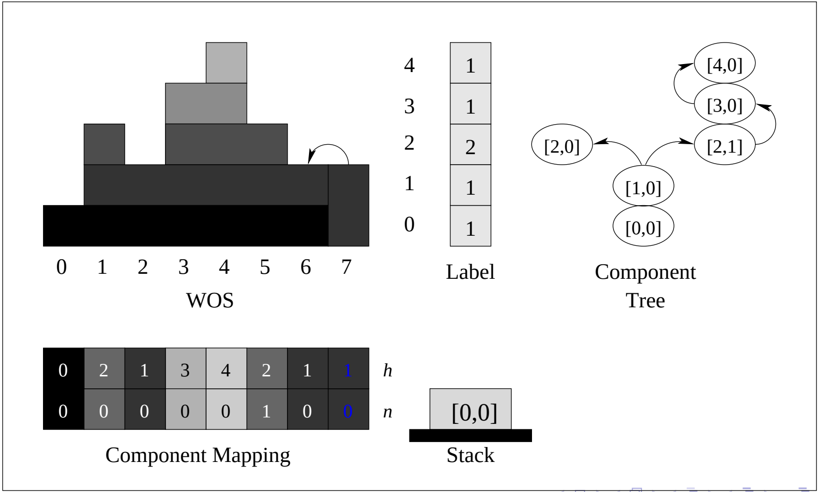
x_6 - $p_h < r_h$ (Ending Point) and $p_h = q_h$ (Stack)



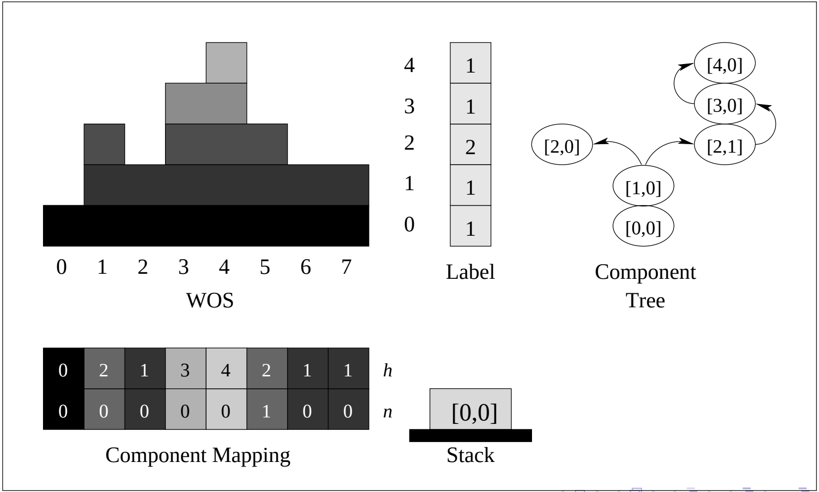
x_7 - $p_h = r_h$ (At the same level)



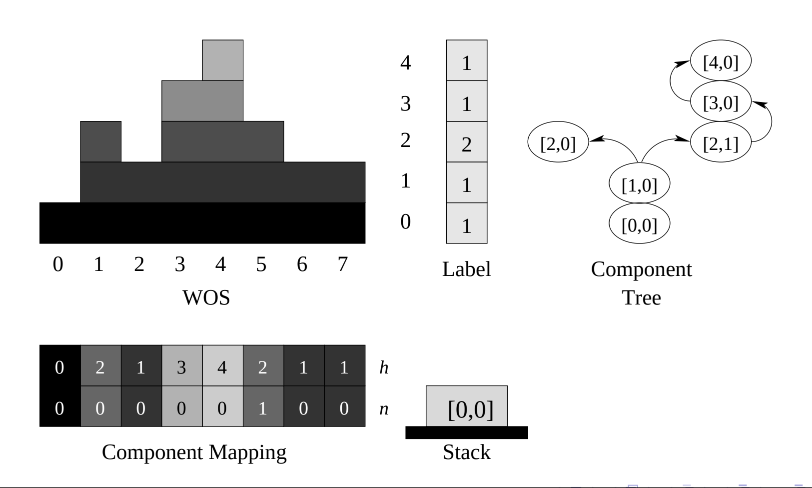
x_7 - p_h = r_h (At the same level)



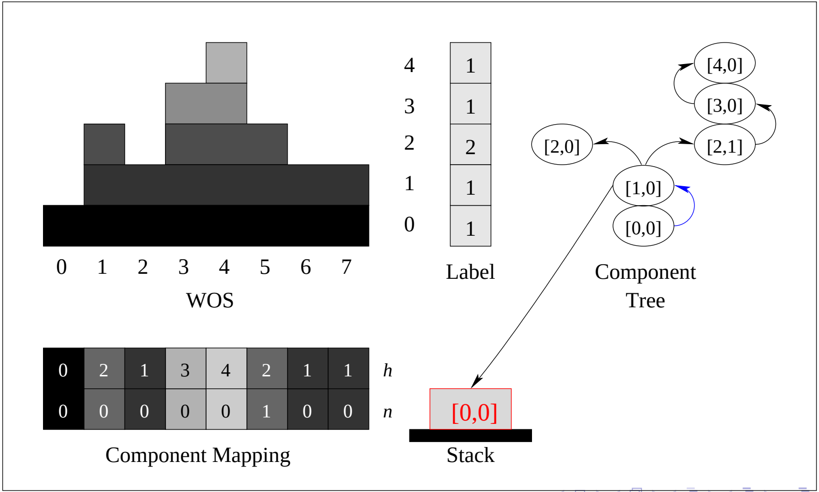
x_7 - p_h = r_h (At the same level)



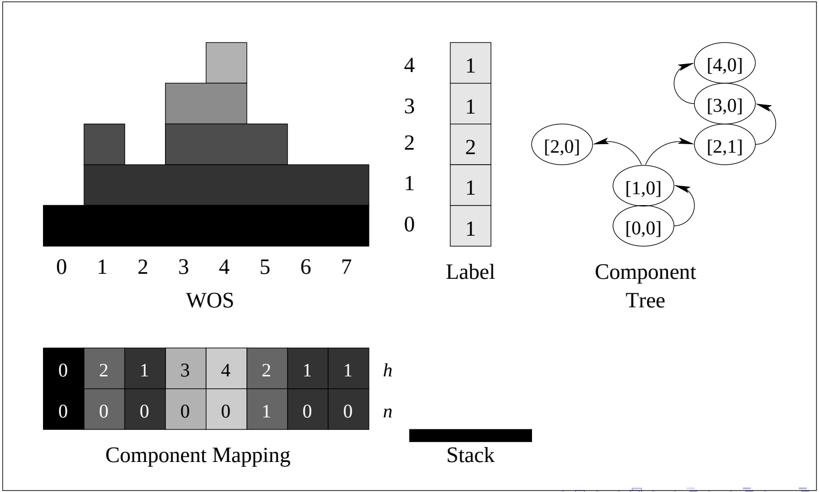
Finishing - Until the Stack Is Empty



Finishing - Until the Stack Is Empty



Finishing - Until the Stack Is Empty



A Possible Implementation

Data: $(P, \prec, F)$ - weighted ordered set with n points
Result: CT - component tree structure
Result: M - a map from P to $[h_{min}...h_{max}, 0...n-1]$

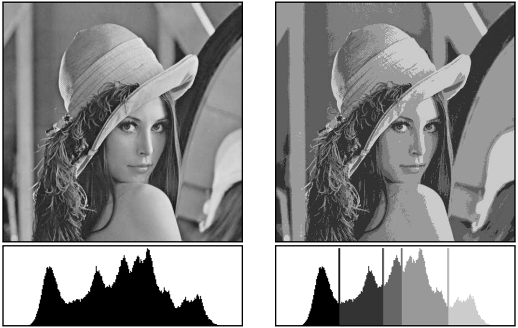
```
// Starting Point - Initialization
for i ← 1 ; i < n ; i++ do // Processing
    if (p_h > r_h) then StackPush(CP, [r_h, r_n]) ;
    else if (p_h = r_h) then // code
    else if (p_h < r_h) then
        while (!StackEmpty(CP)) do
            [q_n, q_h] ← StackView(CP);
            if (p_h ≥ q_h) then break;
            StackPop(CP) ;
        if (StackEmpty(CP) and (p_h < r_h)) or (p_h > q_h) then // code
        else if (p_h = q_h) then StackPop(CP) ;
while (!StackEmpty(CP)) do StackPop(CP) ; // Finishing
```

Time and Space Complexity

- Space - $O(max(m,n))$
  - n - number of points in the WOS
    - the maximum stack size
  - m - number of levels/weight, *i.e.*, $h_{max} - h_{min} + 1$ (*e.g.*, $L = 256$)
    - a vector for the current label at each level h
- Time $O(max(m,n))$
  - Initialization (label vector) $O(m)$
  - Processing $n-1$ points, *i.e.*, $O(n-1)$
    - The component pointed by a point is inserted into the stack only once (worst case)
  - Finalizing $n-1$ points (worst case), *i.e.*, $O(n-1)$

Segmentation by Multiple-threshold Selection

- Histogram Clustering/Classification



Segmentation by Multiple-threshold Selection

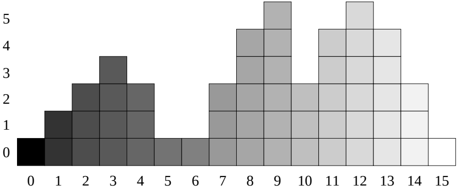
- Assumption - homogeneous regions present in the image can be detected in the histogram of the image
- Five main steps
  1. Histogram Computation
  2. Computation of the Component Tree
  3. Identification of Saliencies [Najman & Couprie, 2006, Algorithm 3]
  4. Histogram Segmentation
  5. Image Segmentation

1) Histogram Computation

Original Image

5	4	4	3	11	13	14
4	3	3	2	11	12	14
3	2	1	10	12	12	13
2	11	11	10	7	8	8
1	12	12	7	8	9	9
0	13	13	8	9	9	9
	0	1	2	3	4	5

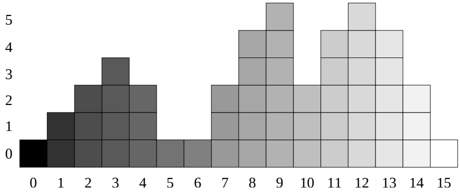
Computed 1D Histogram

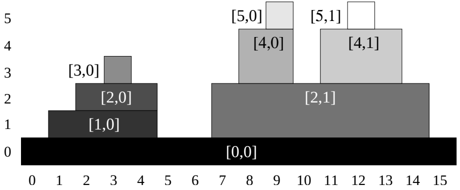


$6 \times 6 = 36$ pixels and $2^4 = 16$ levels

2) Computation of the Component Tree

Original Histogram

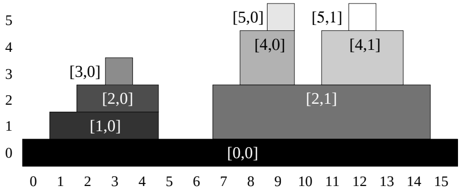


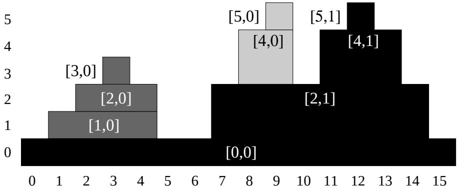


The Component Tree

3) Identification of Saliencies

The Component Tree

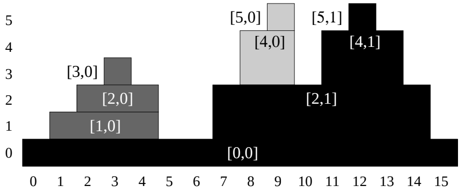


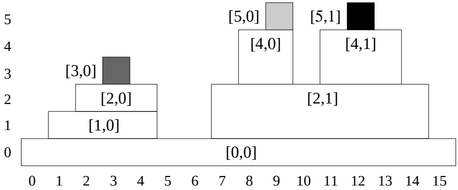


Saliencies

4) Histogram Segmentation - Watershed 1/3

Saliencies

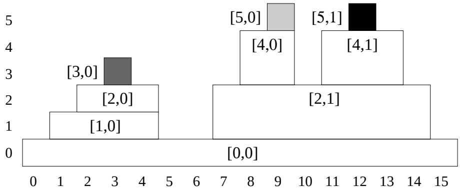


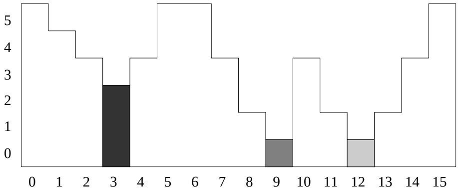


Markers

4) Histogram Segmentation - Watershed 2/3

Markers

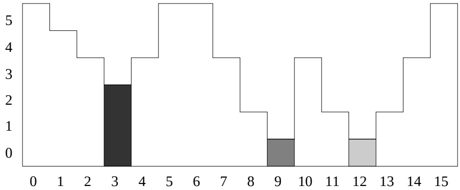


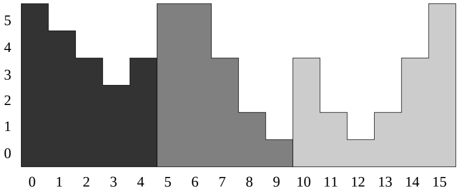


Minima

4) Histogram Segmentation - Watershed 3/3

Minima

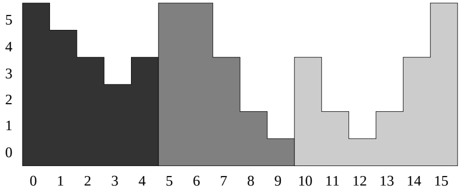


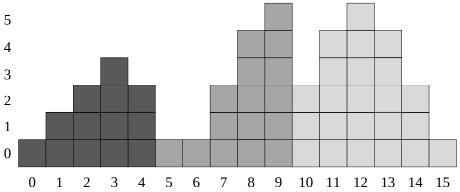


Segmented
Inverse Histogram

5) Image Segmentation 1/2

Segmented
Inverse Histogram

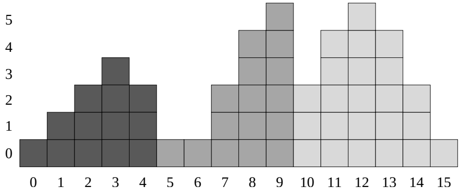




Segmented
Original Histogram

5) Image Segmentation 2/2

Segmented
Original Histogram



5	4	4	3	11	13	14
4	3	3	2	11	12	14
3	2	1	10	12	12	13
2	11	11	10	7	8	8
1	12	12	7	8	9	9
0	13	13	8	9	9	9
	0	1	2	3	4	5

Input × Output
Image

5	3	3	3	12	12	12
4	3	3	3	12	12	12
3	3	3	12	12	12	12
2	12	12	12	8	8	8
1	12	12	8	8	8	8
0	12	12	8	8	8	8
	0	1	2	3	4	5

Input / Output

	0	1	2	3	4	5
5	4	4	3	11	13	14
4	3	3	2	11	12	14
3	2	1	10	12	12	13
2	11	11	10	7	8	8
1	12	12	7	8	9	9
0	13	13	8	9	9	9

	0	1	2	3	4	5
5	3	3	3	12	12	12
4	3	3	3	12	12	12
3	3	3	12	12	12	12
2	12	12	12	8	8	8
1	12	12	8	8	8	8
0	12	12	8	8	8	8

Image Lena - 5 regions

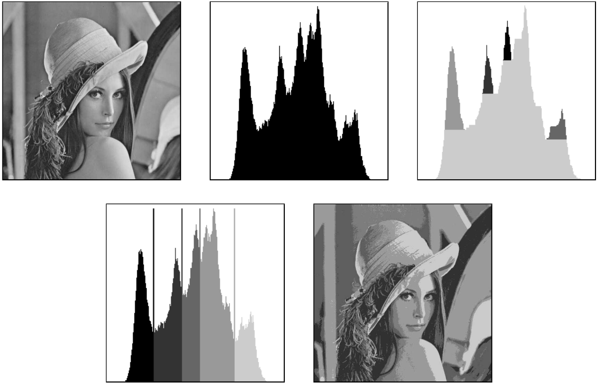


Image House - 5 regions

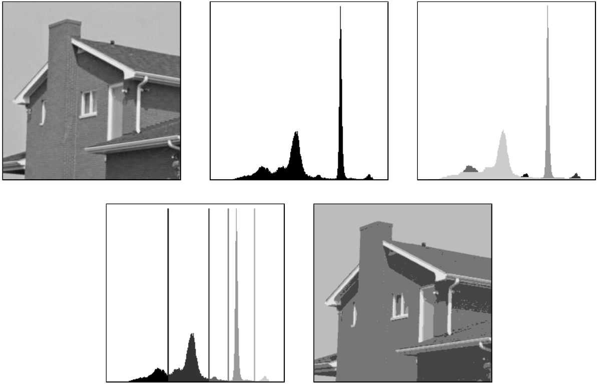
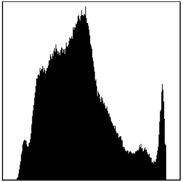
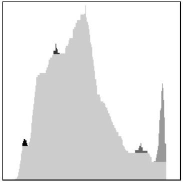
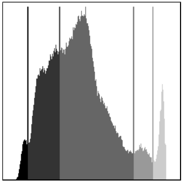


Image GoldHill - 5 regions


PSNR for test images

Images	Kapur	Khotanzad	Our Method	Otsu **Optimal**
lena	25.3574	27.0722	27.5316	28.2001
goldhill	21.8978	22.4819	21.6181	27.0583
fruits	20.7996	22.5991	19.6554	26.3987
barbara	25.4540	26.1957	26.4002	27.1348
cameraman	19.3428	25.5831	25.2907	27.8837
house	20.1270	28.2576	28.1030	29.3351

Conclusion

- A (easy to implement) time and space linear complexity algorithm to compute the Component Tree for 1D signals
- A new method for multithresholding gray-level images
  - Hypothesis - objects that appear on an image can be represented by salient classes present in a histogram of the image
  - Salient classes were modelled as the most significative components (volume attribute)
  - Experiments showed that our method is competitive to classical ones when the hypothesis hold

Future Work

- Methodology to select automatically the number of the most significative components present in the component tree
  - yielding an automatic multithresholding algorithm with respect to the number of classes in the output image
- Improve the way to select the most significative components
- Extend our method to segment color images [Geraud et al., 2001]
- Application to
  - Image Contrast Enhancement through Histogram Equalization
  - Automatic Gray-Level Range Selection on Medical Images

Questions

That's all folks!

Thanks for your attention!

Questions?

References 1/3

JONES, R.
Component trees for image filtering and segmentation.
In: IEEE Workshop on Nonlinear Signal and Image Processing. *Proceedings...* 1997.

SALEMBIER, P. and OLIVERAS, A. and GARRIDO L.
Anti-extensive connected operators for image and sequence processing.
IEEE Transaction on Image Processing. 1998, vol. 4, no. 7, pp. 555–570.

MATTES, J. and RICHARD, M. and DEMONGEOT, J.
Tree representation for image matching and object recognition.
In: 8th International Conference on Discrete Geometry for Computer Imagery, *Proc.*, LNCS-1568, 1999, pp. 298–312.

BREEN, E. J. and JONES, R.
Attribute openings, thinnings and granulometries.
Computer Vision and Image Understading. 1996, vol. 3, no. 64, pp. 377–389.

References 2/3

MATTES, J. and DEMONGEOT, J.
Efficient algorithms to implement the confinement tree.
In: 9th International Conference on Discrete Geometry for Computer Imagery, *Proc.*, LNCS-1953, 2000, pp. 392–405.

NAJMAN, L. and COUPRIE, M.
Building the component tree in quasi-linear time.
IEEE Transaction on Image Processing. 2006, vol. 15, no. 11, pp. 3531–3539.

BEUCHER, S. and MEYER, F.
The morphological approach to segmentation: The watershed transform.
In: Mathematical Morphology in Image Processing. 1992, pp. 433–481.

CROFT, L. H. and ROBINSON, J. A.
Subband image coding using watershed and watercourse lines of the wavelet transform.
IEEE Transactions on Image Processing. 1994, vol. 3, no. 6, pp. 759–772.

References 3/3

KAPUR, J. and SAHOO, P. and WONG, A.
A new method for gray-level picture thresholding using the entropy of the histogram.
Computer Vision, Graphics, and Image Processing. 1985, vol. 29, pp. 273–285.

KHOTANZAD, A. and BOUARFA, A.
Image segmentation by a parallel, non-parametric histogram based clustering algorithm.
Pattern Recognition. 1990, vol. 23, no. 9, pp. 961–973.

OTSU, N.
A threshold selection method from grey-level histograms.
IEEE Transactions on Systems, Man and Cybernetics. 1979, vol. 9, no. 1, pp. 41–47.

GERAUD, T. and STRUB, P.-Y. and DARBON, J.
Color Image Segmentation based on Automatic Morphological Clustering
In: *IEEE* International Conference of Image Processing. 2001, vol. 3, pp. 70–73.

RABBANI, M. and JONES, P. W.
Digital Image Compression Techniques.
1st ed. Bellingham, WA, USA : Society of Photo-Optical Instrumentation Engineers (SPIE), 1991.