Effective Component Tree Computation with Application to Pattern Recognition in Astronomical Imaging

Christophe Berger[1] Thierry Géraud[1] Roland Levillain[1]
Nicolas Widynski[1] Anthony Baillard[2] Emmanuel Bertin[2]

[1]EPITA Research and Development Laboratory (LRDE), Paris, France

[2]Institut d'Astrophysique de Paris (IAP), France

International Conference on Image Processing (ICIP)
September 18, 2007

Effective Component Tree Computation with Application to Pattern Recognition in Astronomical Imaging

Context

- Goal: apply connected filters from mathematical morphology to astronomical images.
- Features of processed astronomical images
  - Huge sizes (order of magnitude: 100 MB – 1.5 GB)
  - Pixels encoded as floating-point values
  - High dynamic ranges

$\Rightarrow$ New tools needed to write these filters, in particular a component tree algorithm

- Joint-work between IAP and LRDE, in the context of the EFIGI project (Extraction of Idealized Patterns of Galaxies in Imaging)

http://www.efigi.org

Context

- Goal: apply connected filters from mathematical morphology to astronomical images.
- Features of processed astronomical images
  - Huge sizes (order of magnitude: 100 MB – 1.5 GB)
  - Pixels encoded as floating-point values
  - High dynamic ranges

$\Rightarrow$ New tools needed to write these filters, in particular a component tree algorithm

- Joint-work between IAP and LRDE, in the context of the EFIGI project (Extraction of Idealized Patterns of Galaxies in Imaging)

http://www.efigi.org

Context

- Goal: apply connected filters from mathematical morphology to astronomical images.
- Features of processed astronomical images
  - Huge sizes (order of magnitude: 100 MB – 1.5 GB)
  - Pixels encoded as floating-point values
  - High dynamic ranges

$\Rightarrow$ New tools needed to write these filters, in particular a component tree algorithm

- Joint-work between IAP and LRDE, in the context of the EFIGI project (Extraction of Idealized Patterns of Galaxies in Imaging)

http://www.efigi.org

Context

- Goal: apply connected filters from mathematical morphology to astronomical images.
- Features of processed astronomical images
  - Huge sizes (order of magnitude: 100 MB – 1.5 GB)
  - Pixels encoded as floating-point values
  - High dynamic ranges

$\Rightarrow$ New tools needed to write these filters, in particular a component tree algorithm

- Joint-work between IAP and LRDE, in the context of the EFIGI project (Extraction of Idealized Patterns of Galaxies in Imaging)

http://www.efigi.org

Context

- Goal: apply connected filters from mathematical morphology to astronomical images.
- Features of processed astronomical images
  - Huge sizes (order of magnitude: 100 MB – 1.5 GB)
  - Pixels encoded as floating-point values
  - High dynamic ranges

$\Rightarrow$ New tools needed to write these filters, in particular a component tree algorithm

- Joint-work between IAP and LRDE, in the context of the EFIGI project (Extraction of Idealized Patterns of Galaxies in Imaging)

http://www.efigi.org

Context

- Goal: apply connected filters from mathematical morphology to astronomical images.
- Features of processed astronomical images
  - Huge sizes (order of magnitude: 100 MB – 1.5 GB)
  - Pixels encoded as floating-point values
  - High dynamic ranges

$\Rightarrow$ New tools needed to write these filters, in particular a component tree algorithm

- Joint-work between IAP and LRDE, in the context of the EFIGI project (Extraction of Idealized Patterns of Galaxies in Imaging)

http://www.efigi.org

Context

- Goal: apply connected filters from mathematical morphology to astronomical images.
- Features of processed astronomical images
  - Huge sizes (order of magnitude: 100 MB – 1.5 GB)
  - Pixels encoded as floating-point values
  - High dynamic ranges

$\Rightarrow$ New tools needed to write these filters, in particular a component tree algorithm

- Joint-work between IAP and LRDE, in the context of the EFIGI project (Extraction of Idealized Patterns of Galaxies in Imaging)

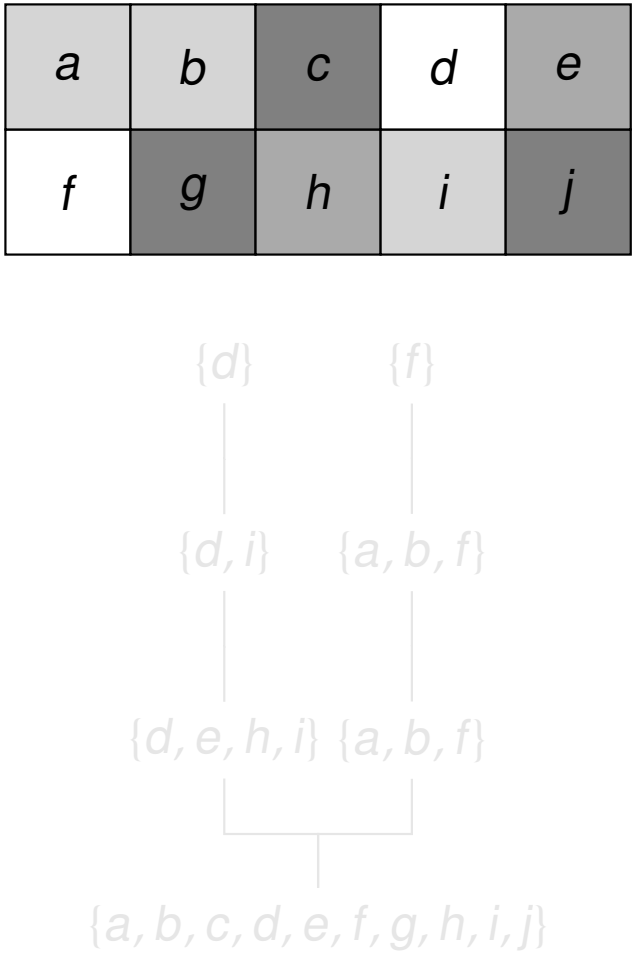
`http://www.efigi.org`

Component Tree

- Convenient and versatile representation of an image
- Parenthood relationship between nodes maps component (spatial) inclusion
- Applications
  - Classification
  - Image Filtering
  - Segmentation
  - Registration
  - Compression

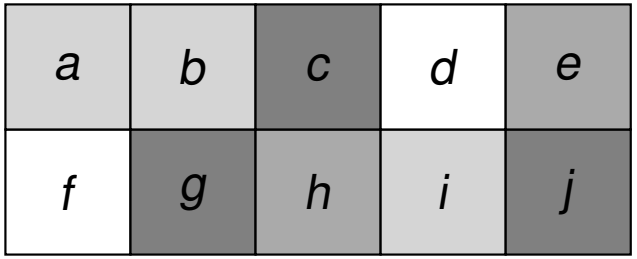
Component Tree

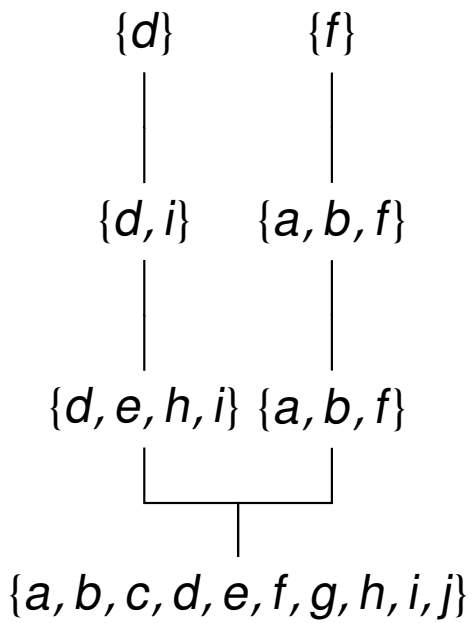
- Convenient and versatile representation of an image
- Parenthood relationship between nodes maps component (spatial) inclusion
- Applications
  - Classification
  - Image Filtering
  - Segmentation
  - Registration
  - Compression



Component Tree

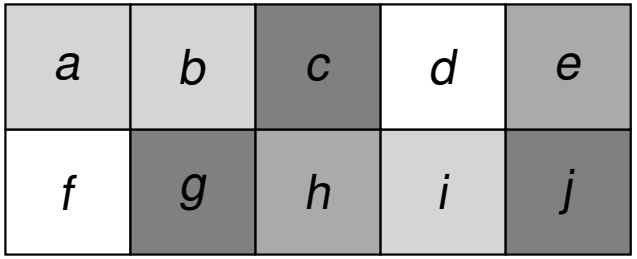
- Convenient and versatile representation of an image
- Parenthood relationship between nodes maps component (spatial) inclusion
- Applications
  - Classification
  - Image Filtering
  - Segmentation
  - Registration
  - Compression

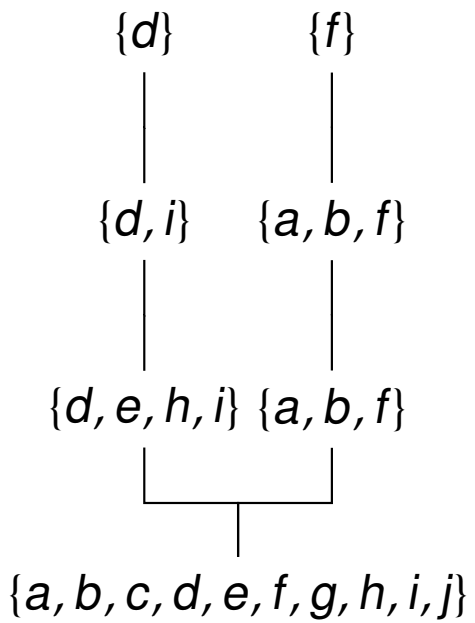




Component Tree

- Convenient and versatile representation of an image
- Parenthood relationship between nodes maps component (spatial) inclusion
- Applications
  - Classification
  - Image Filtering
  - Segmentation
  - Registration
  - Compression





Connected Filters

- Properties
  - Rely on attributes of components (no structuring element)
  - Simplify the images
  - Do not create nor shift contours
- Relationship with the component tree
  - A connected filter can be expressed as a transformation of the component tree that does not add any branch
- Recent filters (about ten years of existence)

Connected Filters

- Properties
  - Rely on attributes of components (no structuring element)
  - Simplify the images
  - Do not create nor shift contours
- Relationship with the component tree
  - A connected filter can be expressed as a transformation on the component tree that does not add any branch
- Recent filters (about ten years of existence)

Connected Filters

- Properties
  - Rely on attributes of components (no structuring element)
  - Simplify the images
  - Do not create nor shift contours
- Relationship with the component tree
  - A connected filter can be expressed as a transformation of the component tree that does not add any branch
- Recent filters (about ten years of existence)

Connected Filters

- Properties
  - Rely on attributes of components (no structuring element)
  - Simplify the images
  - Do not create nor shift contours
- Relationship with the component tree
  - A connected filter can be expressed as a transformation on the component tree that does not add any branch.
- Recent filters (about ten years of existence)

Connected Filters

- Properties
  - Rely on attributes of components (no structuring element)
  - Simplify the images
  - Do not create nor shift contours
- Relationship with the component tree
  - A connected filter can be expressed as a transformation on the component tree that does not add any branch.
- Recent filters (about ten years of existence)

Connected Filters

- Properties
  - Rely on attributes of components (no structuring element)
  - Simplify the images
  - Do not create nor shift contours
- Relationship with the component tree
  - A connected filter can be expressed as a transformation on the component tree that does not add any branch.
- Recent filters (about ten years of existence)

Connected Filters

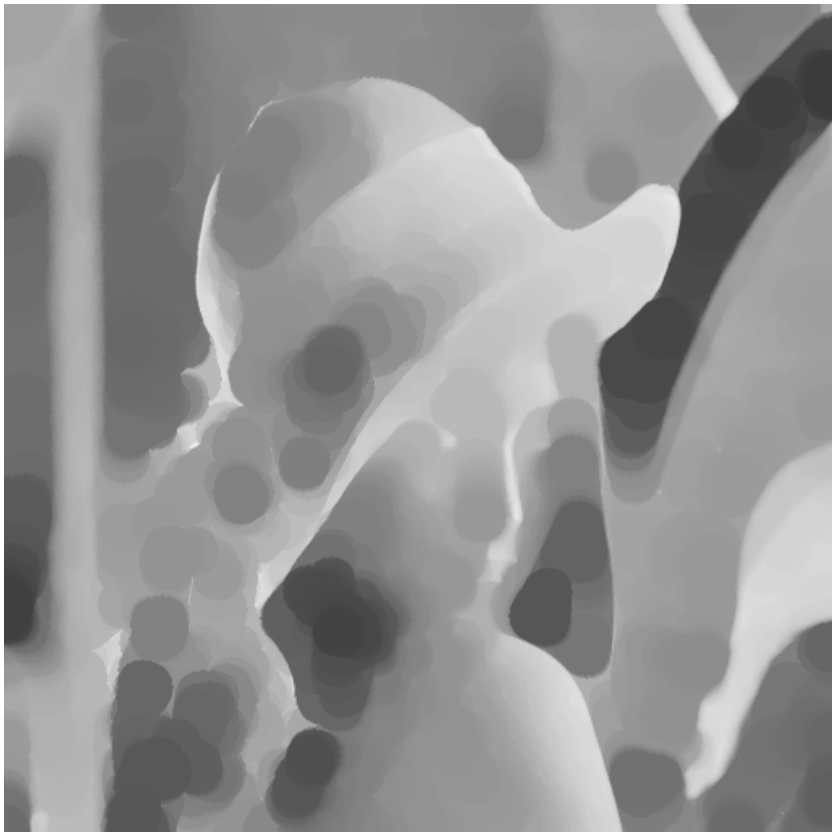
- Properties
  - Rely on attributes of components (no structuring element)
  - Simplify the images
  - Do not create nor shift contours
- Relationship with the component tree
  - A connected filter can be expressed as a transformation on the component tree that does not add any branch.
- Recent filters (about ten years of existence)

Example 1/4

Morphological Opening Using a Structural Element (disc, radius = 15 pixels)

Example 2/4

Morphological Closing Using a Structural Element (disc, radius = 15 pixels)



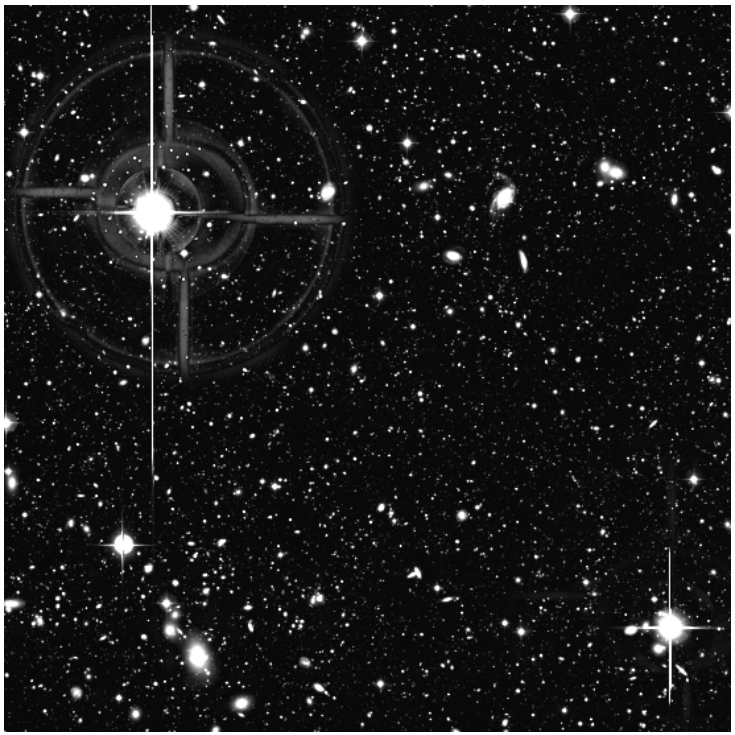
Example 3/4

Morphological Area (attribute) opening (area $\approx \pi 15^2$ pixels)

Example 4/4

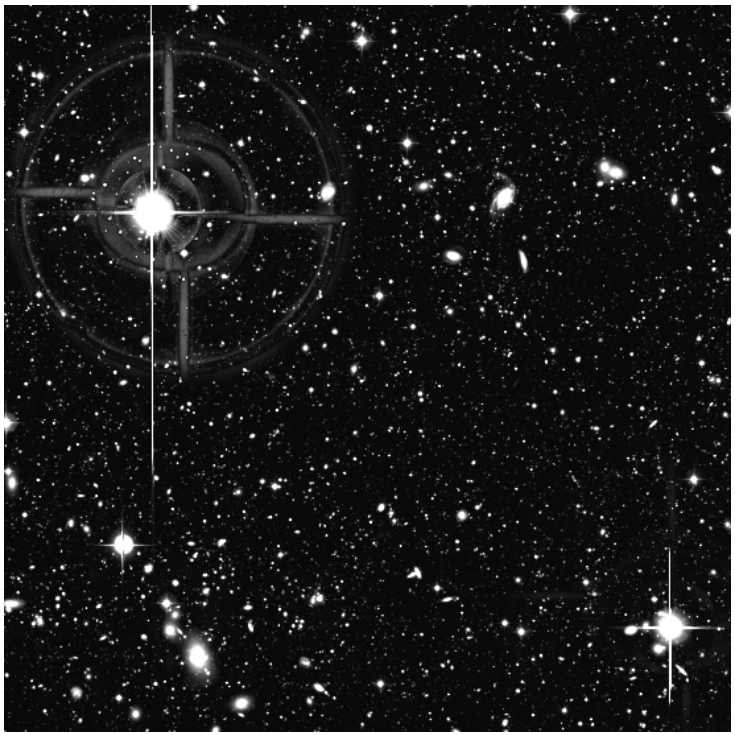
Morphological Area (attribute) closing (area $\approx \pi 15^2$ pixels)

Acquisition and Nature of Data



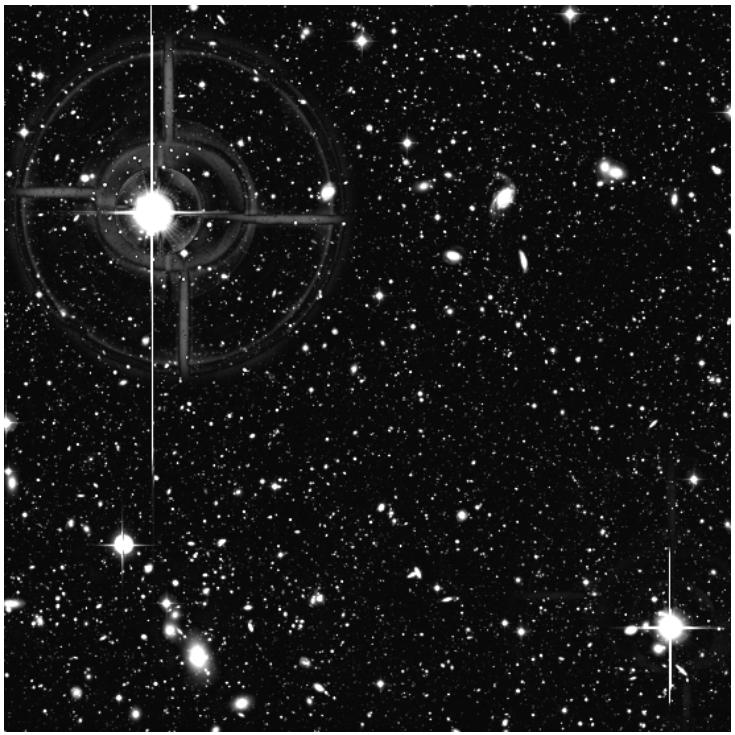
- The observed image is convolved by a Point-Spread Function
- 32-bit, floating-point values
- Most pixels correspond to the (noisy) sky background (dark areas)
- Brighter pixels: objects (stars, galaxies) and optical effects (halos, etc.).

Acquisition and Nature of Data



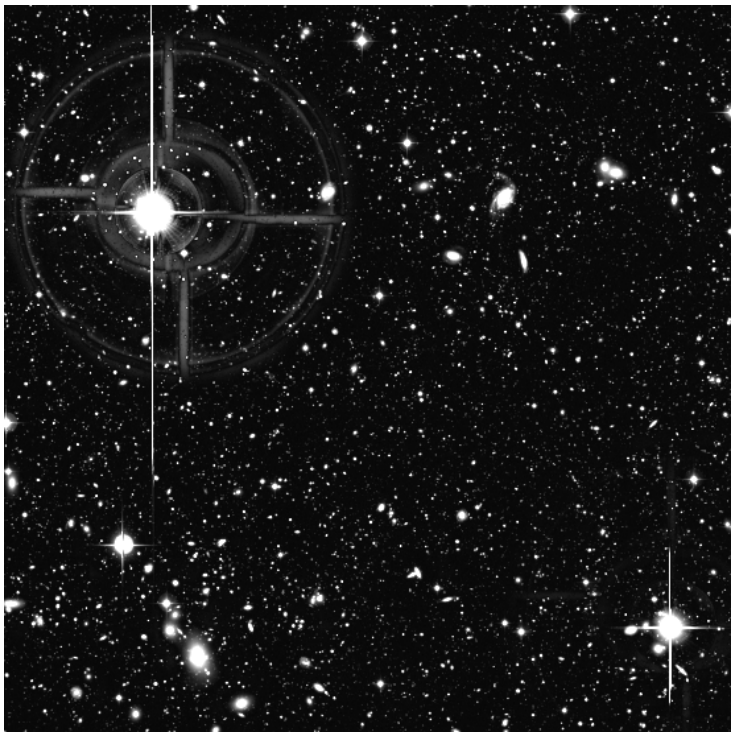
- The observed image is convolved by a Point-Spread Function
- 32-bit, floating-point values
- Most pixels correspond to the (noisy) sky background (dark areas)
- Brighter pixels: objects (stars, galaxies) and optical effects (halos, etc.).

Acquisition and Nature of Data



- The observed image is convolved by a Point-Spread Function
- 32-bit, floating-point values
- Most pixels correspond to the (noisy) sky background (dark areas)
- Brighter pixels: objects (stars, galaxies) and optical effects (halos, etc.).

Acquisition and Nature of Data

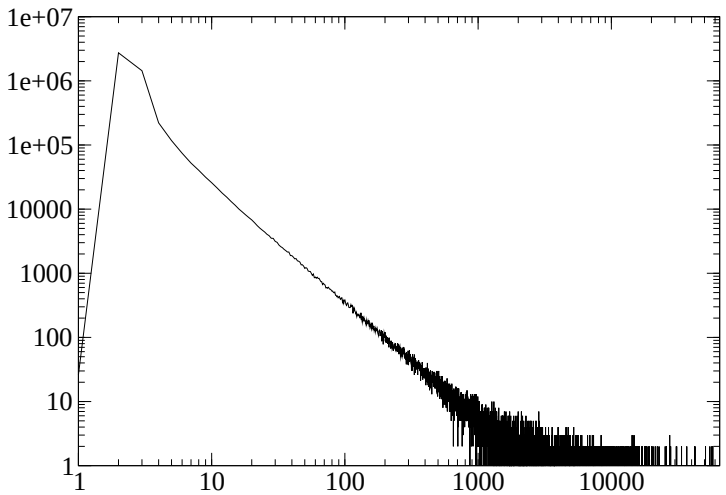


- The observed image is convolved by a Point-Spread Function
- 32-bit, floating-point values
- Most pixels correspond to the (noisy) sky background (dark areas)
- Brighter pixels: objects (stars, galaxies) and optical effects (halos, etc.).

Quantization

Example

Histogram
(*log-log* scale)



- 16-bit, linear quantization
- Most pixels between 0 and 255
- The slope that appears on this plot is due to the presence of blur.

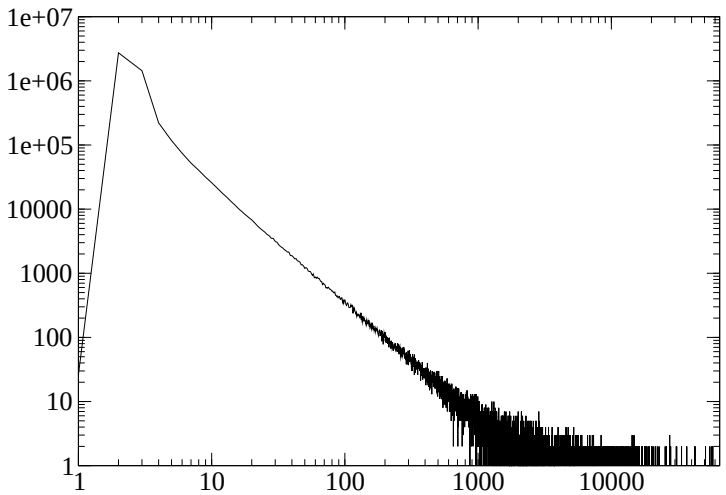
⇒ Need for an optimal quantization; or

⇒ direct processing of floating-point values with no quantization

Quantization

Example

Histogram
(*log-log* scale)



- 16-bit, linear quantization
- Most pixels between 0 and 255
- The slope that appears on this plot is due to the presence of blur.

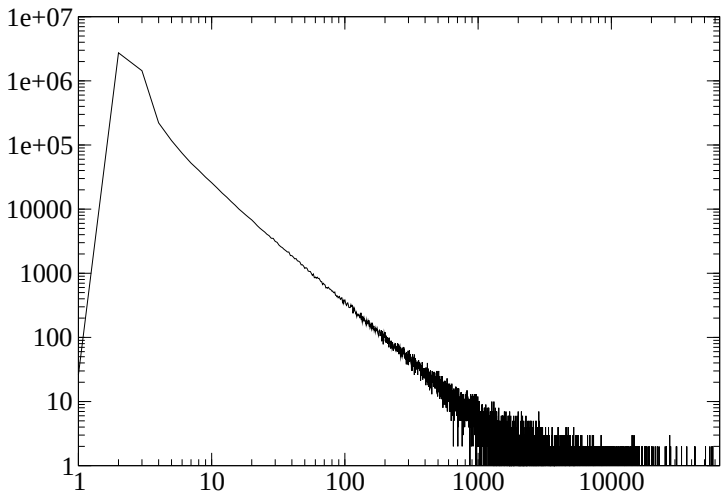
$\Rightarrow$ Need for an optimal quantization; or

$\Rightarrow$ direct processing of floating-point values with no quantization

Quantization

Example

Histogram
(*log-log* scale)



- 16-bit, linear quantization
- Most pixels between 0 and 255
- The slope that appears on this plot is due to the presence of blur.

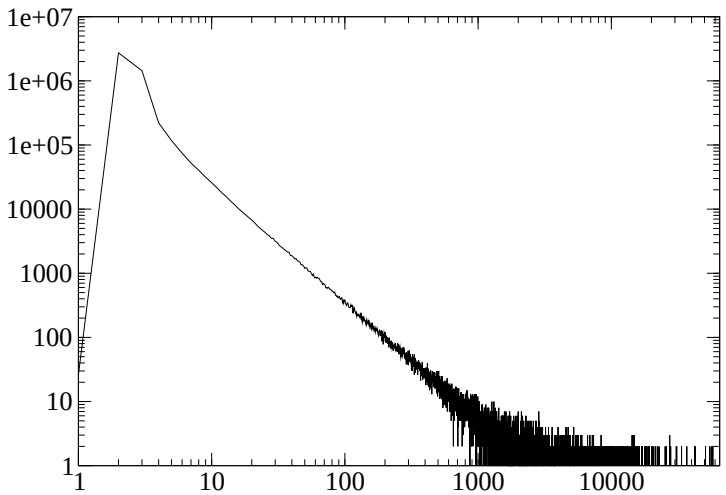
$\Rightarrow$ Need for an optimal quantization; or

$\Rightarrow$ direct processing of floating-point values with no quantization

Quantization

Example

Histogram
(*log-log* scale)



- 16-bit, linear quantization
- Most pixels between 0 and 255
- The slope that appears on this plot is due to the presence of blur.

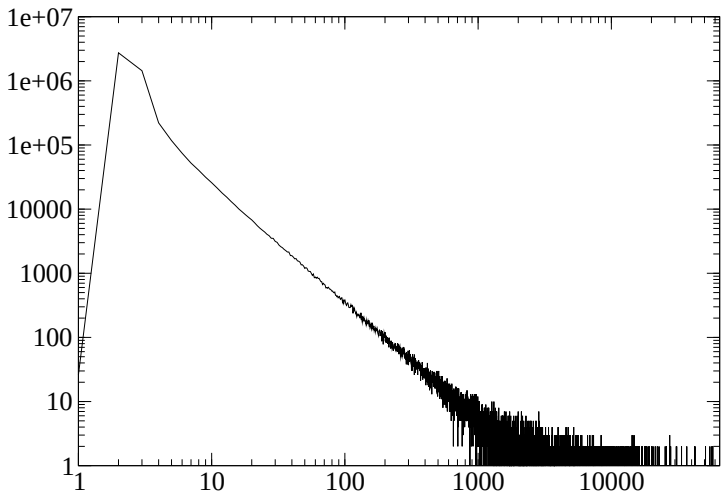
$\Rightarrow$ Need for an optimal quantization; or

$\Rightarrow$ direct processing of floating-point values with no quantization

Quantization

Example

Histogram
(*log-log* scale)



- 16-bit, linear quantization
- Most pixels between 0 and 255
- The slope that appears on this plot is due to the presence of blur.

$\Rightarrow$ Need for an optimal quantization; or

$\Rightarrow$ direct processing of floating-point values with no quantization

Overview

- Based upon a variant of the Union-Find algorithm [Tarjan, 1975]
- Three-step strategy
  1. Ordering of the sites of the image f following a relationship $\mathcal{R}$ such as

$$p \mathcal{R} q \Leftrightarrow \begin{cases} f(p) > f(q), \text{ or} \\ f(p) = f(q) \text{ and } p \text{ is before } q \text{ in the classical video scan order} \end{cases}$$

  2. Actual computation of the component tree
  3. Canonization (compression)

Overview

- Based upon a variant of the Union-Find algorithm [Tarjan, 1975]
- Three-step strategy
  1. Ordering of the sites of the image f following a relationship $\mathcal{R}$ such as

$$p \mathcal{R} q \Leftrightarrow \begin{cases} f(p) > f(q), \text{ or} \\ f(p) = f(q) \text{ and } p \text{ is before } q \text{ in the classical video scan order} \end{cases}$$

  2. Actual computation of the component tree
  3. Canonization (compression)

Overview

- Based upon a variant of the Union-Find algorithm [Tarjan, 1975]
- Three-step strategy
  1. Ordering of the sites of the image f following a relationship $\mathcal{R}$ such as

$$p \mathcal{R} q \Leftrightarrow \begin{cases} f(p) > f(q), \text{ or} \\ f(p) = f(q) \text{ and } p \text{ is before } q \text{ in the classical video scan order} \end{cases}$$

  2. Actual computation of the component tree
  3. Canonization (compression)

Overview

- Based upon a variant of the Union-Find algorithm [Tarjan, 1975]
- Three-step strategy
  1. Ordering of the sites of the image f following a relationship $\mathcal{R}$ such as

$$p \mathcal{R} q \Leftrightarrow \begin{cases} f(p) > f(q), \text{ or} \\ f(p) = f(q) \text{ and } p \text{ is before } q \text{ in the classical video scan order} \end{cases}$$

  2. Actual computation of the component tree
  3. Canonization (compression)

Overview

- Based upon a variant of the Union-Find algorithm [Tarjan, 1975]
- Three-step strategy
  1. Ordering of the sites of the image f following a relationship $\mathcal{R}$ such as

$$p \mathcal{R} q \Leftrightarrow \begin{cases} f(p) > f(q), \text{ or} \\ f(p) = f(q) \text{ and } p \text{ is before } q \text{ in the classical video scan order} \end{cases}$$

  2. Actual computation of the component tree
  3. Canonization (compression)

Component tree and *max-tree*

3	3	1	4	2
4	1	2	3	1

f

C	D	H	A	F
B	I	G	E	J

R

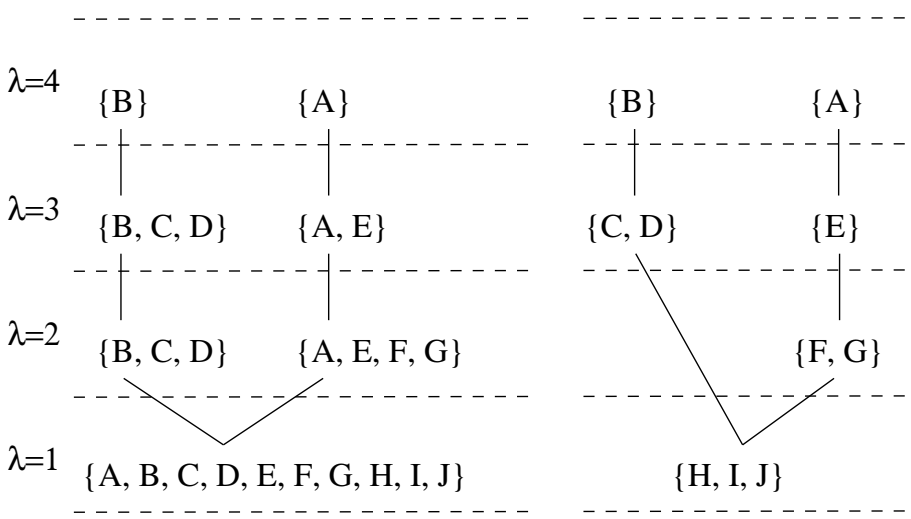


Image Level Sets

3	3	1	4	2
4	1	2	3	1

f

Image Level Sets

3	3	1	4	2
4	1	2	3	1

f

C	D	H	A	F
B	I	G	E	J

R

Image Level Sets

3	3	1	4	2
4	1	2	3	1

f

Level 4 ($\lambda = 4$)

C	D	H	A	F
B	I	G	E	J

R

$\lambda = 4$

Component tree

Max-tree

Image Level Sets

3	3	1	4	2
4	1	2	3	1

f

			A	

Level 4 ($\lambda = 4$)

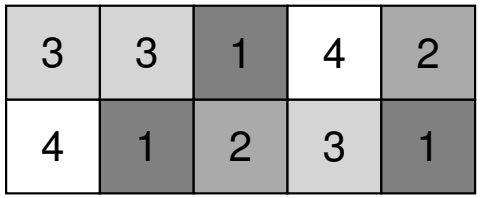
C	D	H	A	F
B	I	G	E	J

R

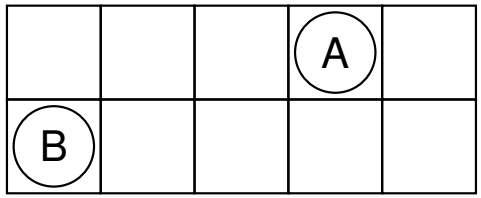
$\lambda = 4$ $\{A\}$ $\{A\}$

Component tree Max-tree

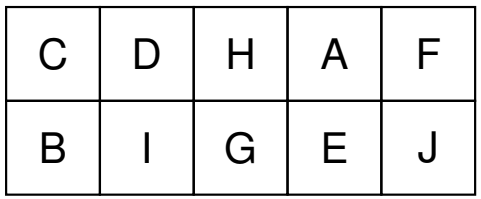
Image Level Sets



f



Level 4 ($\lambda = 4$)



R

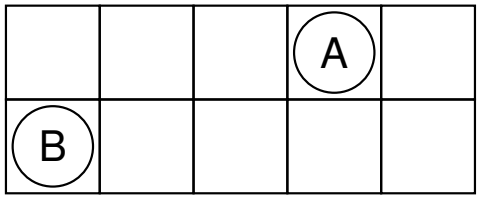
$\lambda = 4$ $\{B\}$ $\{A\}$ $\{B\}$ $\{A\}$

Component tree Max-tree

Image Level Sets

3	3	1	4	2
4	1	2	3	1

f



Level 4 ($\lambda = 4$)

Level 3 ($\lambda = 3$)

C	D	H	A	F
B	I	G	E	J

R

$\lambda = 4$ $\{B\}$ $\{A\}$ $\{B\}$ $\{A\}$

$\lambda = 3$ $\{B\}$ $\{A\}$

Component tree

Max-tree

Image Level Sets

3	3	1	4	2
4	1	2	3	1

f

			A	
B				

Level 4 ($\lambda = 4$)

C			A	
B				

Level 3 ($\lambda = 3$)

C	D	H	A	F
B	I	G	E	J

R

$\lambda = 4$ $\{B\}$ $\{A\}$ $\{B\}$ $\{A\}$

$\lambda = 3$ $\{B, C\}$ $\{A\}$ $\{C\}$

Component tree Max-tree

Image Level Sets

3	3	1	4	2
4	1	2	3	1

f

			A	
B				

Level 4 ($\lambda = 4$)

C	D		A	
B				

Level 3 ($\lambda = 3$)

C	D	H	A	F
B	I	G	E	J

R

$\lambda = 4$ $\{B\}$ $\{A\}$ $\{B\}$ $\{A\}$

$\lambda = 3$ $\{B, C, D\}$ $\{A\}$ $\{C, D\}$

Component tree Max-tree

Image Level Sets

3	3	1	4	2
4	1	2	3	1

f

			A	
B				

Level 4 ($\lambda = 4$)

C	D		A	
B			E	

Level 3 ($\lambda = 3$)

C	D	H	A	F
B	I	G	E	J

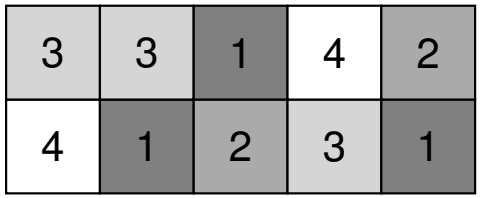
R

$\lambda = 4$ $\{B\}$ $\{A\}$ $\{B\}$ $\{A\}$

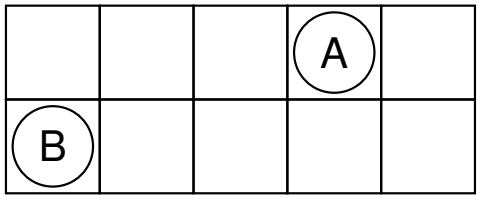
$\lambda = 3$ $\{B, C, D\}$ $\{A, E\}$ $\{C, D\}$ $\{E\}$

Component tree Max-tree

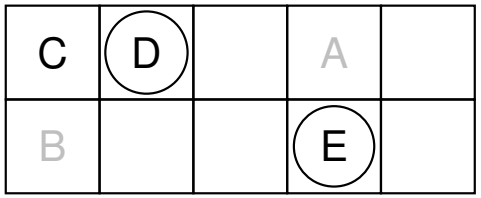
Image Level Sets



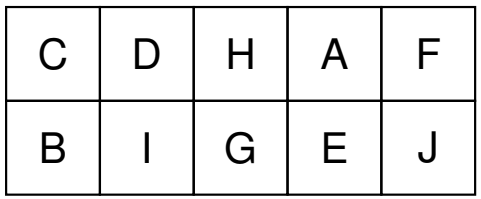
f



Level 4 ($\lambda = 4$)

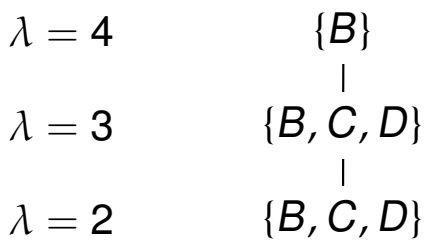


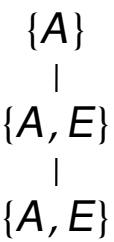
Level 3 ($\lambda = 3$)



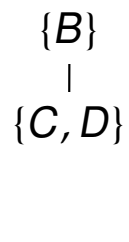
R

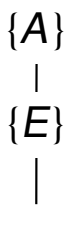
Level 2 ($\lambda = 2$)





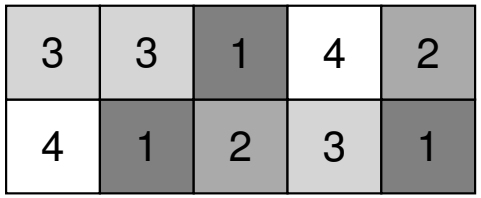
Component tree



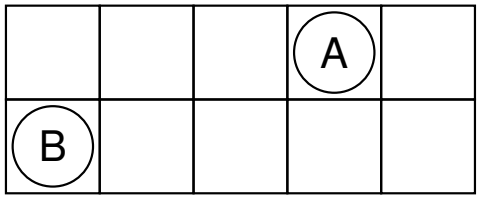


Max-tree

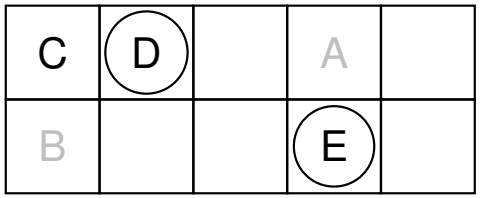
Image Level Sets



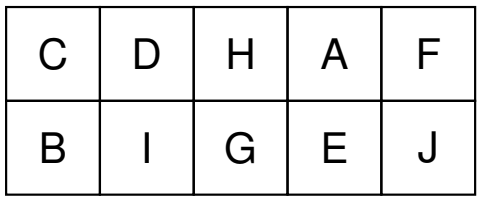
f



Level 4 ($\lambda = 4$)

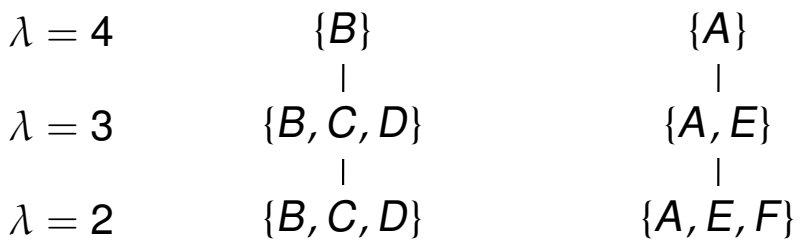


Level 3 ($\lambda = 3$)

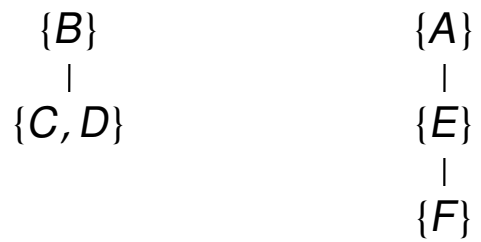


R

Level 2 ($\lambda = 2$)

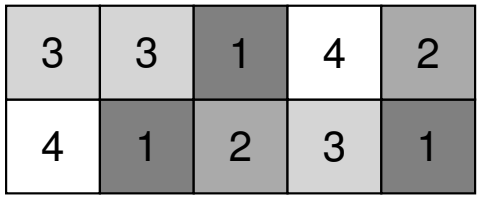


Component tree

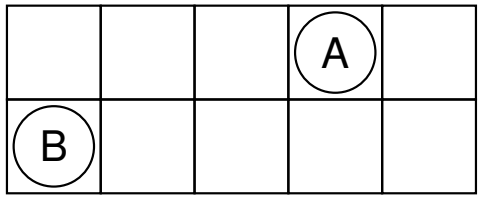


Max-tree

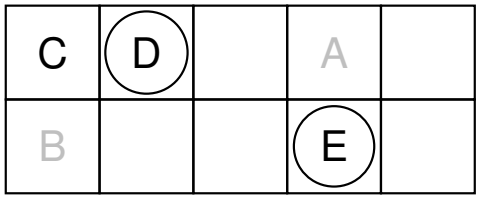
Image Level Sets



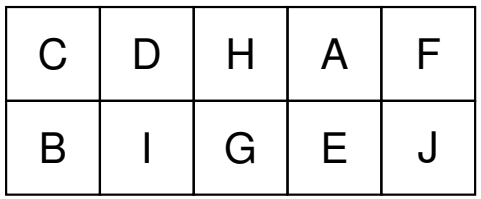
f



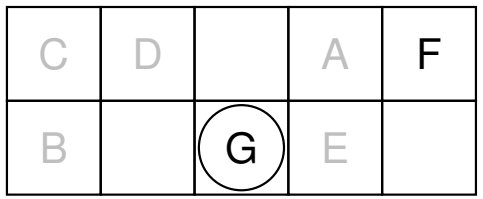
Level 4 ($\lambda = 4$)



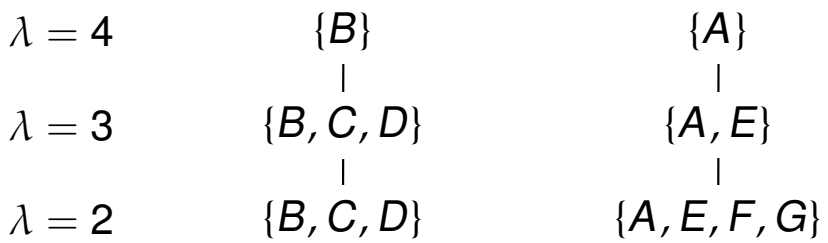
Level 3 ($\lambda = 3$)



R



Level 2 ($\lambda = 2$)



Component tree

Max-tree

Image Level Sets

3	3	1	4	2
4	1	2	3	1

f

			A	
B				

Level 4 ($\lambda = 4$)

C	D		A	
B			E	

Level 3 ($\lambda = 3$)

C	D	H	A	F
B	I	G	E	J

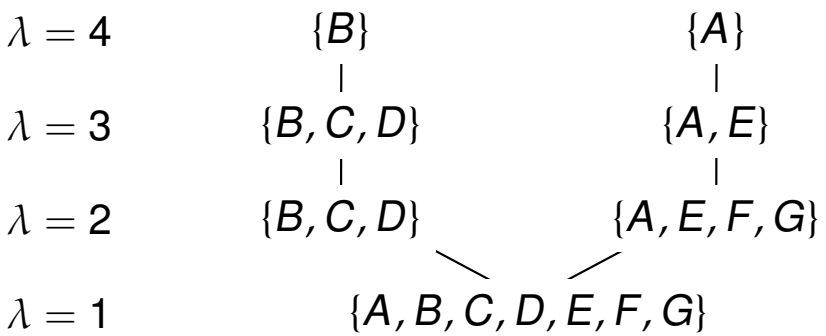
R

C	D		A	F
B		G	E	

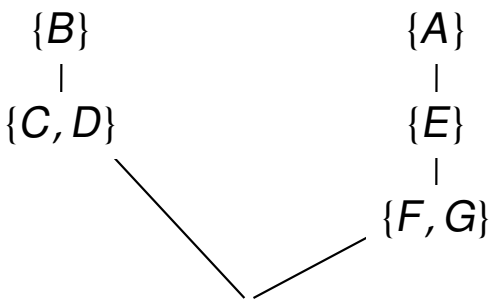
Level 2 ($\lambda = 2$)

C	D		A	F
B		G	E	

Level 1 ($\lambda = 1$)

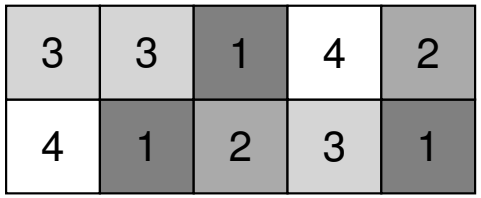


Component tree

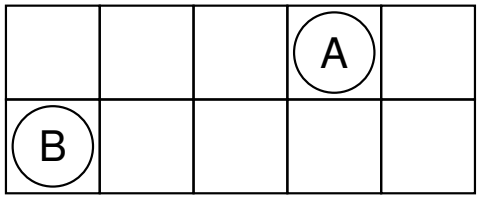


Max-tree

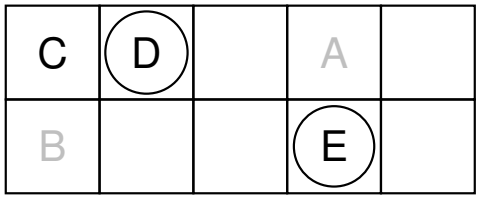
Image Level Sets



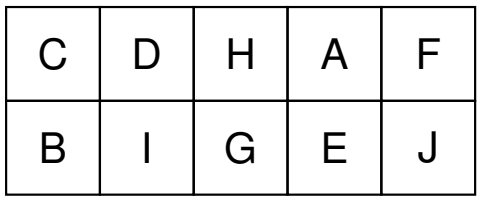
f



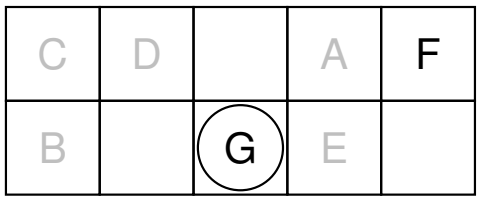
Level 4 ($\lambda = 4$)



Level 3 ($\lambda = 3$)

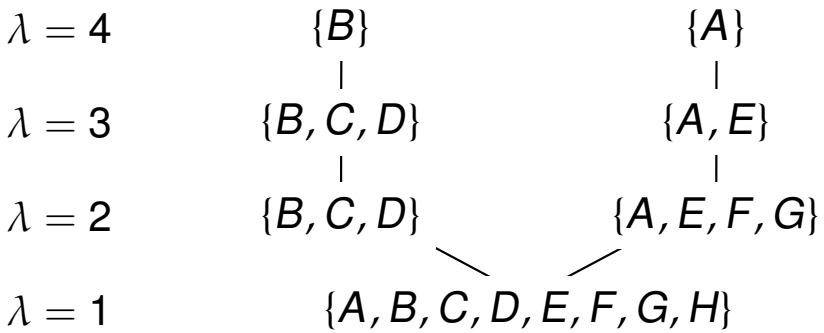


R

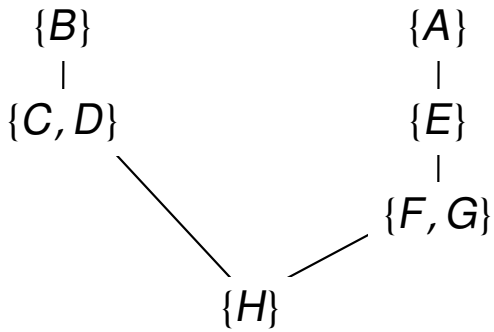


Level 2 ($\lambda = 2$)

Level 1 ($\lambda = 1$)

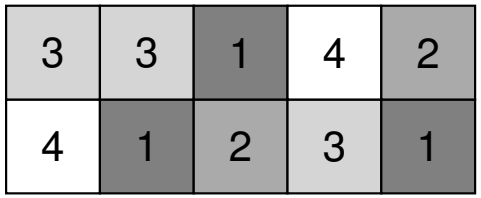


Component tree

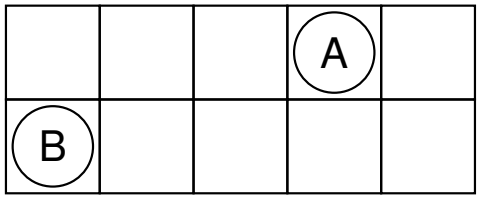


Max-tree

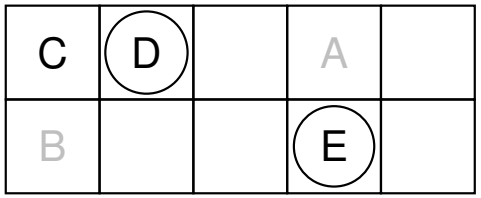
Image Level Sets



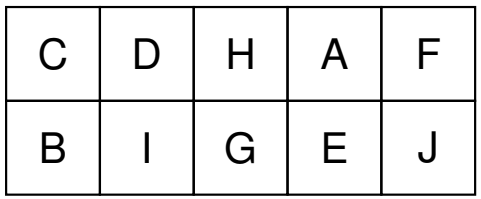
f



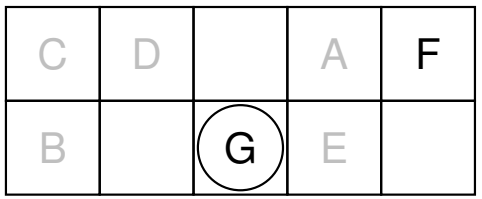
Level 4 ($\lambda = 4$)



Level 3 ($\lambda = 3$)

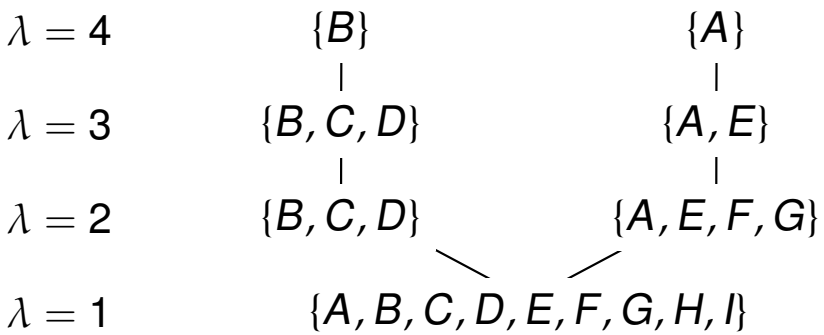


R

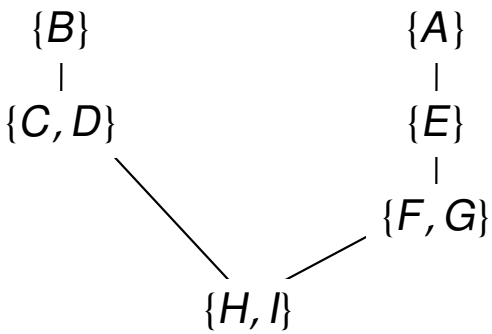


Level 2 ($\lambda = 2$)

Level 1 ($\lambda = 1$)

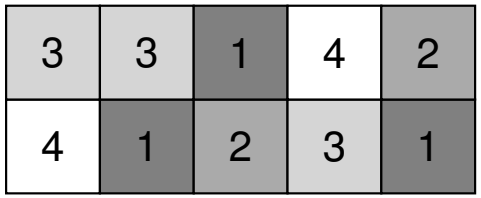


Component tree

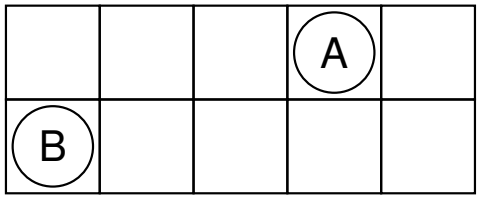


Max-tree

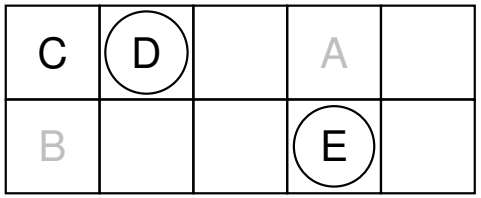
Image Level Sets



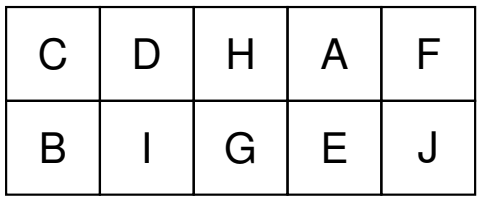
f



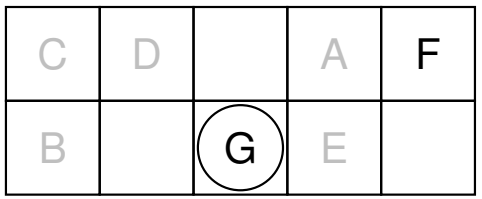
Level 4 ($\lambda = 4$)



Level 3 ($\lambda = 3$)

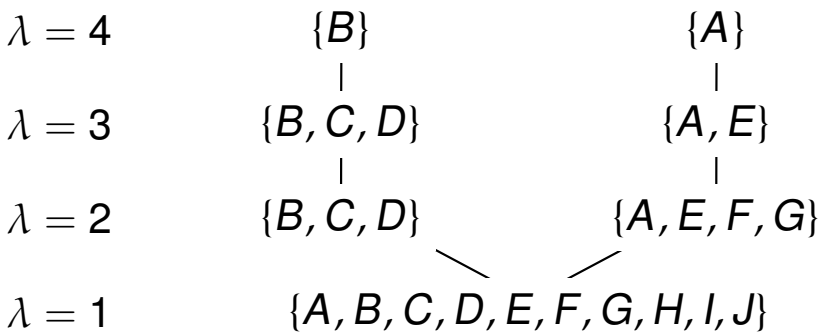


R

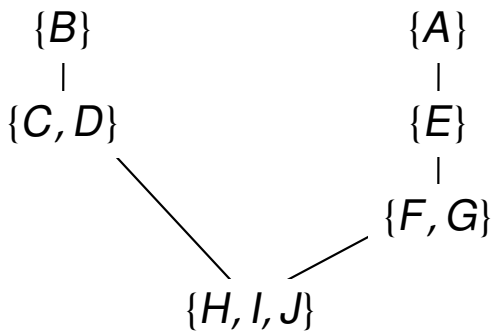


Level 2 ($\lambda = 2$)

Level 1 ($\lambda = 1$)



Component tree



Max-tree

Density of the max-tree, canonization

3	3	1	4	2
4	1	2	3	1

f

C	D	H	A	F
B	I	G	E	J

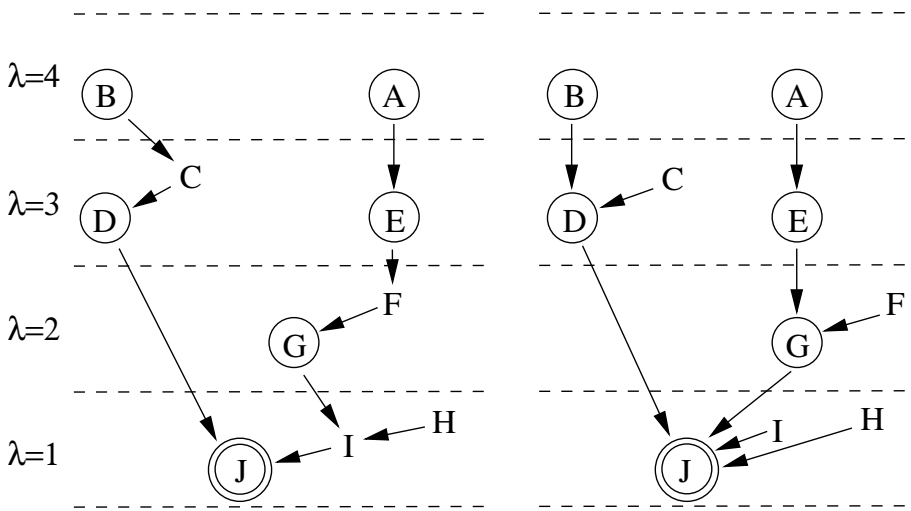
R

D	J	I	E	G
C	J	I	F	J

parent

D	J	J	E	G
D	J	J	G	J

CANONIZE-TREE(*parent*)



Computation

COMPUTE-TREE

FIND-ROOT(x)

```
1  if zpar(x) = x then return x
2    else { zpar(x) ← FIND-ROOT(zpar(x)) ; return zpar(x) }
```

COMPUTE-TREE(f)

```
1  for each p, zpar(p) ← undef
2  R ← REVERSE-SORT(f)   // maps R into an array
3  for each p ∈ R in direct order
4    parent(p) ← p ; zpar(p) ← p
5    for each n ∈ N(p) such as zpar(n) ≠ undef
6      r ← FIND-ROOT(n)
7      if r ≠ p then { parent(r) ← p ; zpar(r) ← p }
8  DEALLOCATE(zpar)
9  return pair(R, parent)   // a "correct" function
```

Canonization

CANONIZE-TREE

CANONIZE-TREE($parent, f$)

1 **for** each $p \in R$ in reverse order
2 $\quad q \leftarrow parent(p)$
3 $\quad$ **if** $f(parent(q)) = f(q)$ **then** $parent(p) \leftarrow parent(q)$
4 **return** *parent* // a "canonized" function

Example

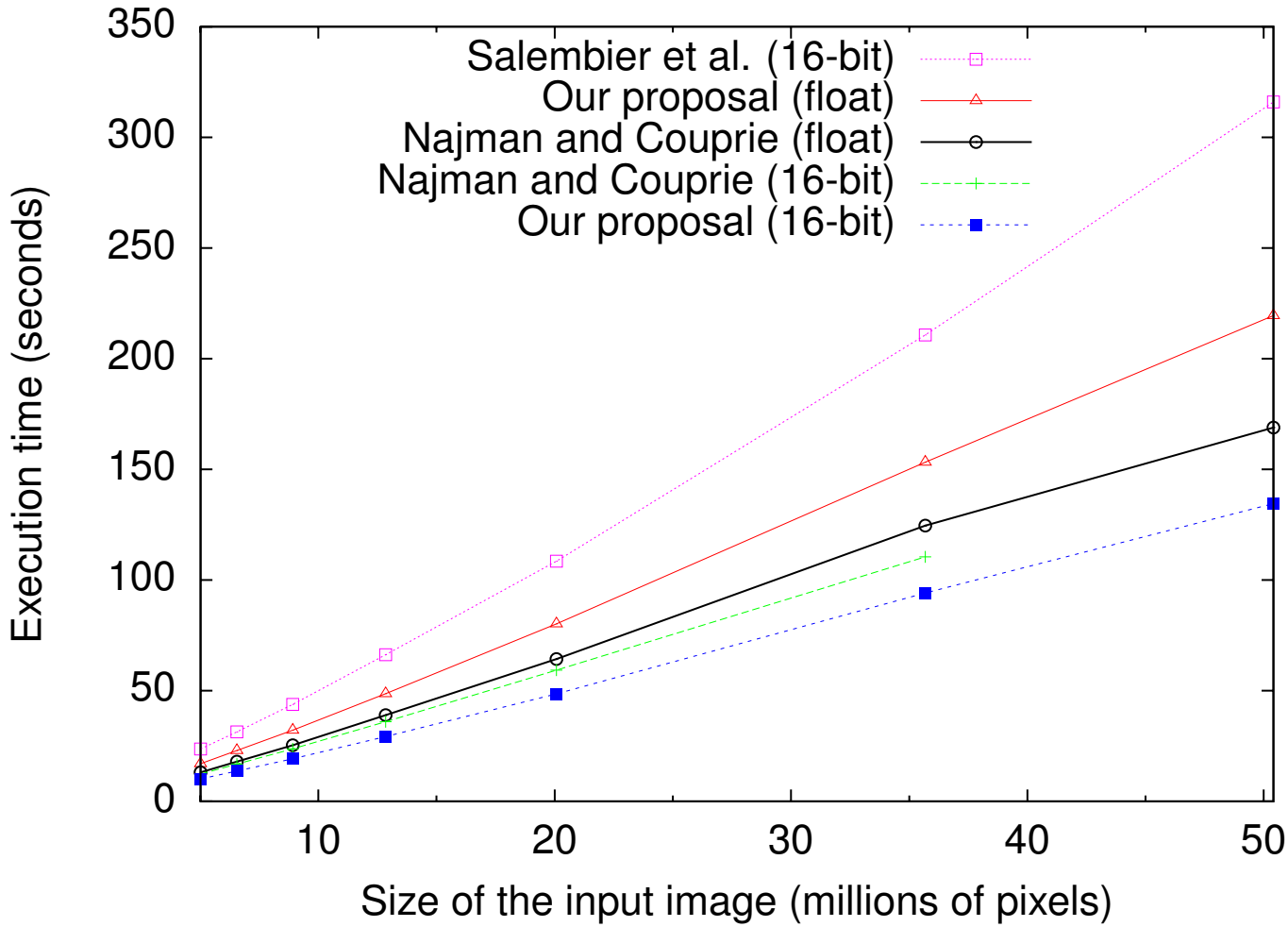
computation of the area of the components

COMPUTE-AREA($f, R, parent$)

1 **for** each $p \in R$, $area(p) \leftarrow 1$ // initialization
2 **for** each $p \in R$ in direct order
3 $area(parent(p)) \leftarrow area(parent(p)) + area(p)$ // update

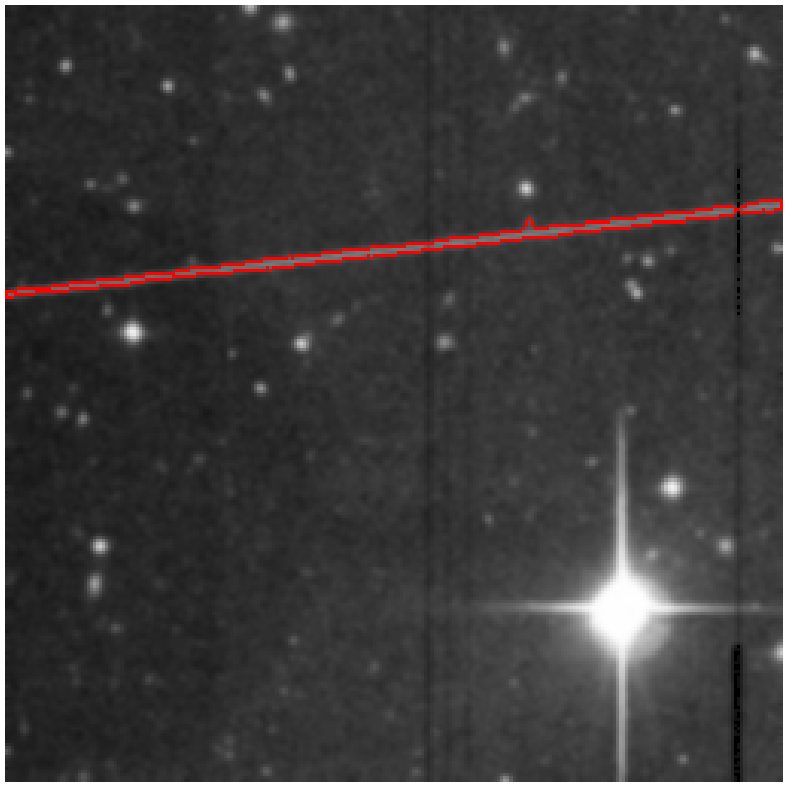
- Simple process
- Computation conducted in an iterative way
- Linear complexity

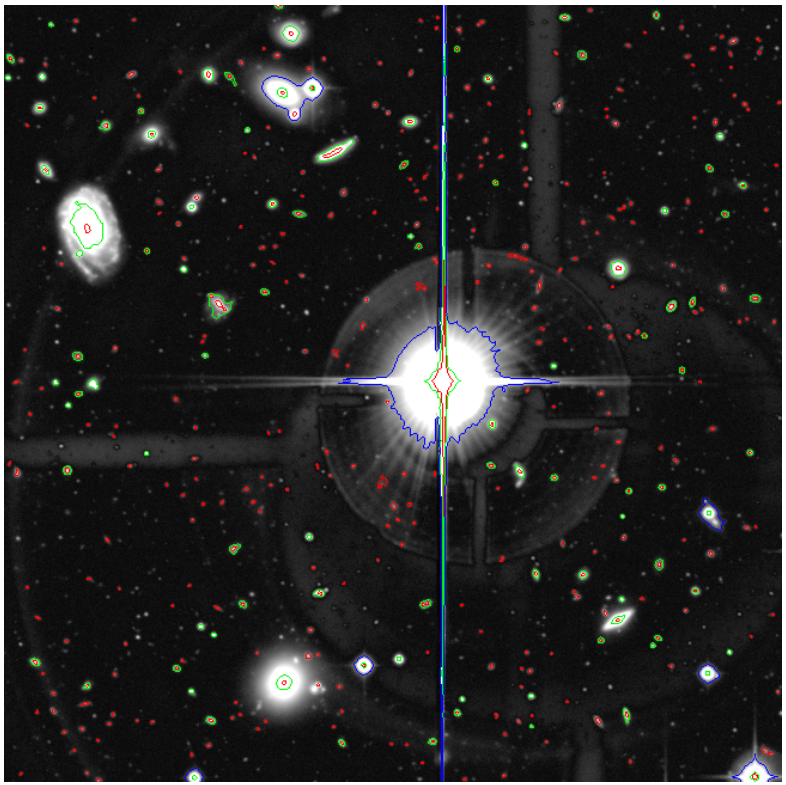
Comparison of Execution Times



(3 Ghz Bi-Xeon Processor with 2×1 MB of cache memory and 4 GB of RAM, running GNU/Linux.)

Applications





Conclusions

- Algorithm effective for images with high quantization and with no quantization
- About as efficient as the fastest known algorithm, and needs twice less memory
- Serves as a basis to build efficient connected filters for astronomical images
- On-going work on the selection of attributes and node labeling strategies

Conclusions

- Algorithm effective for images with high quantization and with no quantization
- About as efficient as the fastest known algorithm, and needs twice less memory
- Serves as a basis to build efficient connected filters for astronomical images
- On-going work on the selection of attributes and node labeling strategies

Conclusions

- Algorithm effective for images with high quantization and with no quantization
- About as efficient as the fastest known algorithm, and needs twice less memory
- Serves as a basis to build efficient connected filters for astronomical images
- On-going work on the selection of attributes and node labeling strategies

Conclusions

- Algorithm effective for images with high quantization and with no quantization
- About as efficient as the fastest known algorithm, and needs twice less memory
- Serves as a basis to build efficient connected filters for astronomical images
- On-going work on the selection of attributes and node labeling strategies

Effective Component Tree Computation with Application to Pattern Recognition in Astronomical Imaging

Bibliography I

Géraud, Th. (2005).
Ruminations on Tarjan's Union-Find algorithm and connected operators.
In *Mathematical Morphology: 40 Years On (Proc. of ISMM)*, volume 30 of *Computational Imaging and Vision*, pages 105–116, Paris, France. Springer.

Jones, R. (1997).
Component trees for image filtering and segmentation.
In Coyle, E., editor, *IEEE Workshop on Nonlinear Signal and Image Processing*, Mackinac Island.

Bibliography II

Meijster, A. (2004).
Efficient Sequential and Parallel Algorithms for Morphological Image Processing.
PhD thesis, University of Groningen, the Netherlands.

Meijster, A. and Wilkinson, M. H. F. (2002).
A comparison of algorithms for connected set openings and closings.
IEEE Trans. Pattern Anal. Machine Intell., 24(4):484–494.

Najman, L. and Couprie, M. (2006).
Building the component tree in quasi-linear time.
IEEE Trans. Image Processing, 15(11):3531–3539.

Bibliography III


Salembier, P., Oliveras, A., and Garrido, L. (1998).
Antiextensive connected operators for image and sequence processing.
IEEE Trans. Image Processing, 7(4):555–570.


Tarjan, R. E. (1975).
Efficiency of a good but not linear set union algorithm.
Journal of the ACM, 22(2):215–225.


Vincent, L. (1993).
Grayscale area openings and closings: their applications and efficient implementation.
In *Intl. Symposium on Mathematical Morphology*, pages 22–27.