university of groningen

Connected Filters

Michael H. F. Wilkinson

Institute for Mathematics and Computing Science
University of Groningen
The Netherlands

SSIP, Vienna, July 11, 2008

Vienna, 1899

Outline

- Filters by Reconstruction
- Attribute filters
- Max-trees:
  - Data structure
  - Use in attribute filtering
  - Shape filters and distributions
  - Computing multi-variate pattern spectra
- Adapting the Max-tree:
  - Iso-surface browsing
  - Splatting
  - Parallel computation
  - Other work

An Example

Lenna with noise (left) structural open-close with square S.E. (middle) area open-close (right)

Definitions I

- Let E be some universal set and $\mathcal{P}(E)$ the set of all subsets of E.
- For a binary image $X \subseteq E$ a *connected foreground component* C is a connected subset of X of maximal extent.
- If C is a connected foreground component of X we denote this as $C \Subset X$.
- A *connected backround component* of X is a connected foreground component of the complement X^c of X.
- A *partition* of E is a set $A \subset \mathcal{P}(E)$ of sets α_i for which

$$\bigcup_i \alpha_i = E, \quad \text{and} \tag{1}$$

$$i \neq j \Rightarrow \alpha_i \cap \alpha_j = \emptyset. \tag{2}$$

- The set A_X of all connected foreground and background components of X form a partition of E.

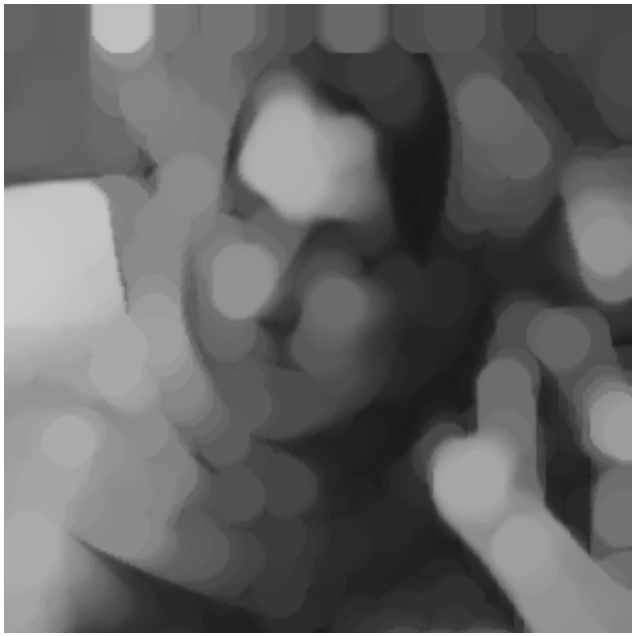
Definitions II

- Let $A = \{\alpha_i\}$ and $B = \{\beta_j\}$ be partitions.
- A is said to be *finer* than B *iff* for every α_i there exists a β_j such that $\alpha_i \subseteq \beta_j$.
- If A is finer than B then B is *coarser* than A.
- Let $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ be a binary image operator, and $A_{\Psi(X)}$ be the partition of E consisting of all foreground and background components of $\Psi(X)$.
- Ψ is a *binary connected operator* if for any image X the partition A_X is finer than $A_{\Psi(X)}$.
- The connected opening Γ_x is defined as

$$\Gamma_x(X) = \begin{cases} C : C \Subset X \wedge x \in C & \text{if } x \in X \\ \emptyset & \text{otherwise.} \end{cases} \tag{3}$$

Reconstruction

original f



marker $g = \gamma_{21} f$

reconstruction of f by g

The edge preserving effect of openings-by-reconstruction compared to structural openings

Reconstruction

- The basis of an opening by reconstruction is the reconstruction of image f from an arbitrary marker g.
- This is usually defined using geodesic dilations $\overline{\delta}_f$ defined as

$$\overline{\delta}_f^1(g) = f \wedge \delta(g). \tag{4}$$

- This operator is used iteratively until stability, to perform the reconstruction ρ i.e.

$$\rho(f|g) = \lim_{n\to\infty} \overline{\delta}_f^n g = \underbrace{\overline{\delta}_f^1 \ldots \overline{\delta}_f^1 \overline{\delta}_f^1}_{\text{until stability}}(g). \tag{5}$$

- In practice we apply $\overline{\delta}_f^n$ with n the smallest integer such that

$$\overline{\delta}_f^n g = \overline{\delta}_f^{n-1} g. \tag{6}$$

Openings-by-Reconstruction

- What this process does in the binary case is reconstruct any connected component in f which intersects some part of g.
- An opening-by-reconstruction $\bar{\gamma}_X$ with structuring element (S.E) X is computed as

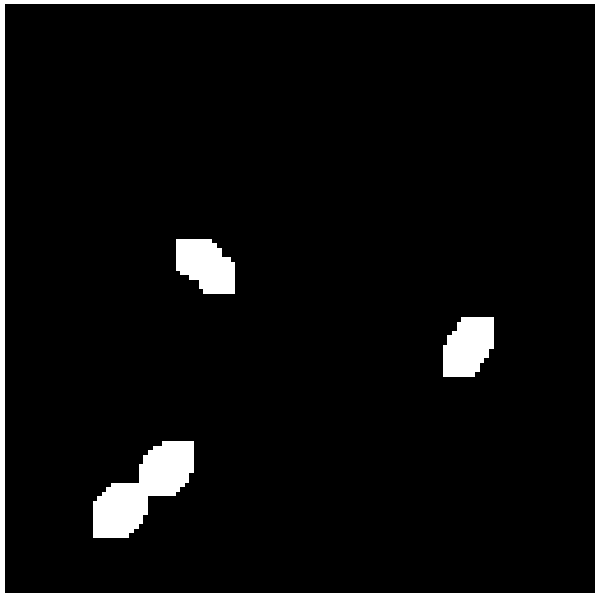
$$\bar{\gamma}_X(f) = \rho(f|\gamma_X(f)), \tag{7}$$

  in which γ_X denotes an opening of f by X.
- Reconstructing from this marker preserves any connected component in which X fits at at least one position.
- Closing-by-reconstruction $\bar{\phi}_X$ can be defined by duality, i.e.

$$\bar{\phi}_X(f) = -\bar{\gamma}_X(-f) \tag{8}$$

Structural Openings vs. Reconstruction

X



$X \circ B$

$\rho(X|X \circ B)$

- The *structural opening* $X \circ B$, with B a 7×7 square, yields the union of all 7×7 squares which fit into X. Clearly this distorts the connected components and is not a connected filter.
- The *opening-by-reconstruction* $\rho(X|X \circ B)$ preserves all connected components of X into which at least one 7×7 square fits. This can be considered an attribute filter.

Levelings

- Openings-by-reconstructions are anti-extensive, and closings-by-reconstructions are extensive, removing bright or dark image details respectively.
- Meyer (*J.Math. Imag. Vis.* 2004) proposed levelings as an auto-dual extension of reconstruction filters.
- In this case a marker is used which may lie partly above and partly below the image.
- We can compute a leveling of $\lambda(f|g)$ of f from marker g as

$$(\lambda(f|g))(x) = \begin{cases} (\rho(f|g))(x) & \text{if } f(x) \geq g(x) \\ -(\rho(-f|-g))(x) & \text{if } f(x) < g(x), \end{cases} \tag{9}$$

- Levelings allow edge-preserving simplification of images, by simultaneously removing bright and dark details.

Example: Levelings

original

blurred by Gaussian

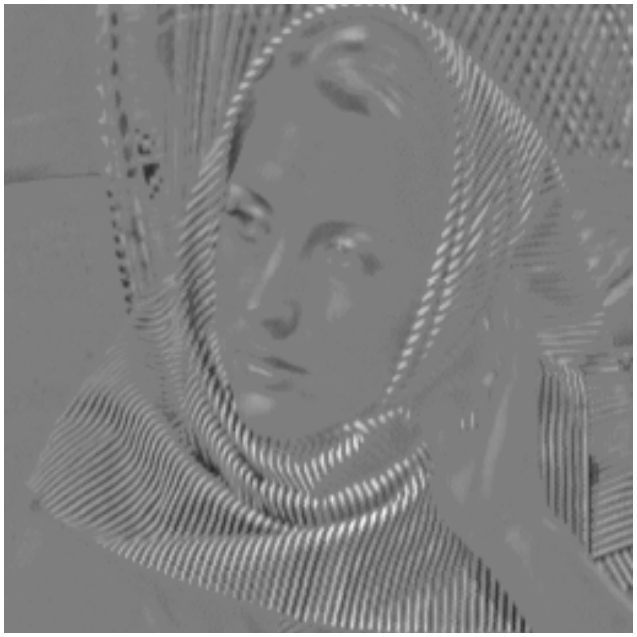
leveling

Leveling using a Gaussian filter to simplify the image in an auto-dual manner.

Example: Leveling Cartoons

original

cartoon



texture channel

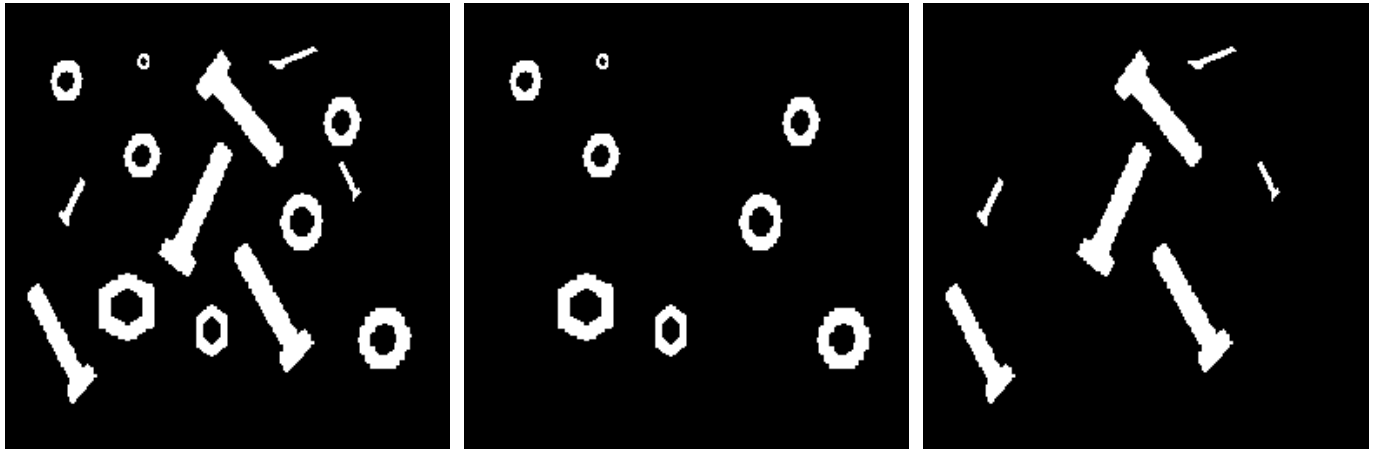
Leveling cartoons for texture/cartoon decomposition.

Attribute Filters I

- Introduced by Breen and Jones in 1996.
- Examples: area openings/closings, attribute openings, shape filters
- How do they work?

Binary image :

1. compute attribute for each connected component
2. keep components of which attribute value exceeds some threshold λ



Attribute Openings: Formally

- Let $T : \mathcal{P}(E) \rightarrow \{false, true\}$ be an increasing criterion, i.e. $C \subseteq D$ implies that $T(C) \Rightarrow T(D)$.
- A binary *trivial opening* $\Gamma_T : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ using T as defined above is defined as

$$\Gamma_T(C) = \begin{cases} C & \text{if } T(C), \\ \emptyset & \text{otherwise.} \end{cases} \tag{10}$$

- A typical form of T is

$$T(C) = (\mu(C) \geq \lambda) \tag{11}$$

in which μ is some increasing scalar attribute value (i.e. $C \subseteq D \Rightarrow \mu(C) \leq \mu(D)$), and λ is the attribute threshold.

- The binary *attribute opening* Γ^T is defined as

$$\Gamma^T(X) = \bigcup_{x \in X} \Gamma_T(\Gamma_x(X)), \tag{12}$$

in other words it is the union of all connected foreground components of X which meet the criterion T.

Attribute Openings: Examples

X

$T = A(C) \geq 11^2$

$T = I(C) \geq 11^4/6$

- An *area opening* is obtained if the criterion $T = A(C) \geq \lambda$, with A the area of the connected set C.
- A *moment-of-inertia opening* is obtained if the criterium is of the form $T = I(C) \geq \lambda$, with I the moment of inertia.

Other Attribute Filters

- If criterion T is non-increasing in (10), Γ_T becomes a *trivial thinning*, or *trivial, anti-extensive grain filter* Φ_T:

$$\Phi_T(C) = \begin{cases} C & \text{if } T(C), \\ \emptyset & \text{otherwise.} \end{cases} \tag{13}$$

- Using a trivial thinning rather than a trivial opening in (12), Γ_T becomes an *attribute thinning* or *anti-extensive grain filter* Φ^T:

$$\Phi^T(X) = \bigcup_{x \in X} \Phi_T(\Gamma_x(X)), \tag{14}$$

- The extensive dual of the atribute opening Γ_T is the *attribute closing* Ψ_T, which is defined as

$$\Psi_T(X) = (\Gamma_T(X^c))^c. \tag{15}$$

- The extensive dual of the attribute thinning is the *attribute thickening*, which is defined as above, but with a non-increasing criterion.

Non-Increasing Attributes

- Attribute thinnings can be defined using the usual form $T(C) = (\mu(C) \geq \lambda)$ if μ is non-increasing, e.g.:
  - Perimeter length P
  - Circularity (or boundary complexity) P^2/A
  - Concavity: $(H - A)/A$, with H the convex hull area
  - Elongation (non-compactness): I/A^2
  - Any of Hu's moment invariants
- Alternatively, increasing attributes (i.e. $C \subseteq D \Rightarrow \mu(C) \leq \mu(D)$) can be used if the form of T is changed:
  - $T = (\mu(C) = \lambda)$
  - $T = (\mu(C) \leq \lambda)$
  - etc.

The Grey-scale Case

- In the case of attribute openings, generalization to grey scale is achieved through *threshold decomposition*.
- A threshold set X_h of grey level image (function) f is defined as

$$X_h(f) = \{x \in E | f(x) \geq h\}. \tag{16}$$

- The grey scale attribute opening γ^T based on binary counterpart Γ^T is given by

$$(\gamma^T(f))(x) = \sup\{h \leq f(x) | x \in \Gamma^T(X_h(f))\} \tag{17}$$

- Closings ψ^T are defined by duality:

$$\psi^T(f) = -\gamma^T(-f). \tag{18}$$

- The non-increasing case will be dealt with after discussing the algorithms.

(Semi) Auto-dual filtering

- A filter is auto-dual (or self-dual) if it is invariant to inversion:

$$\psi(f) = -\psi(-f) \tag{19}$$

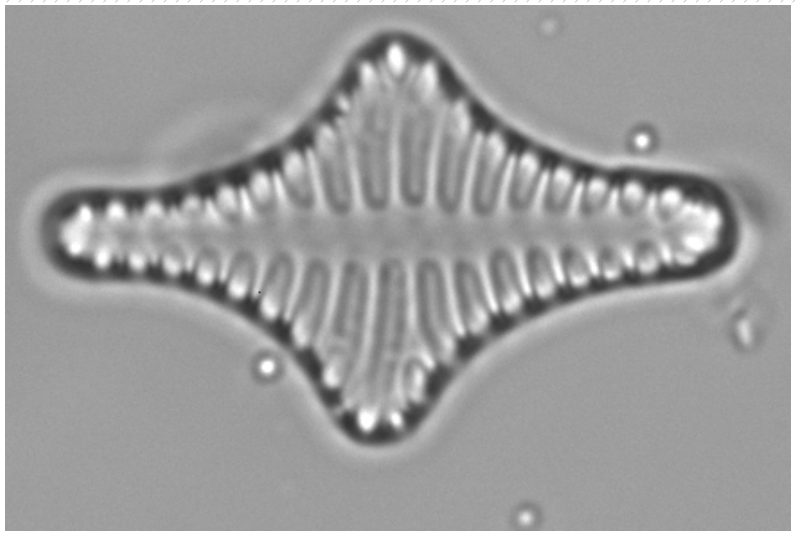
- An approximation is offered by *alternating sequential filters* (ASFs), which consist of an alternating sequence of openings and closings of increasing scale (e.g. radius of structuring element).
- Let γ_λ^a be a area opening of attribute threshold λ, and ϕ_λ^a the corresponding area closing.
- The *area* N-Sieve ψ_λ^N is given by

$$\psi_\lambda^N(f) = \phi_\lambda^a(\gamma_\lambda^a(\ldots(\phi_2^a(\gamma_2^a(\phi_1^a(\gamma_1^a(f)))))\ldots)) \tag{20}$$

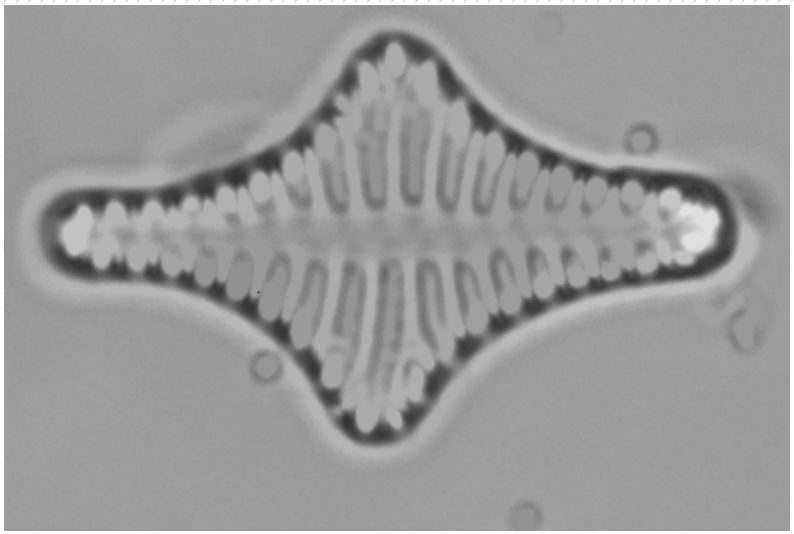
  and is an alternating sequential filter.
- The corresponding M-Sieve ψ_λ^M is just

$$\psi_\lambda^M(f) = \gamma_\lambda^a(\phi_\lambda^a(\ldots(\gamma_2^a(\phi_2^a(\gamma_1^a(\phi_1^a(f)))))\ldots)) \tag{21}$$

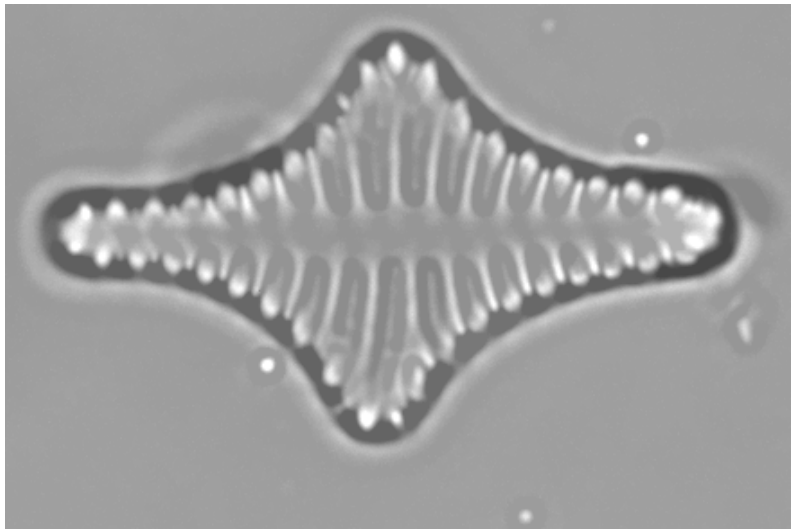
Grey Scale Example



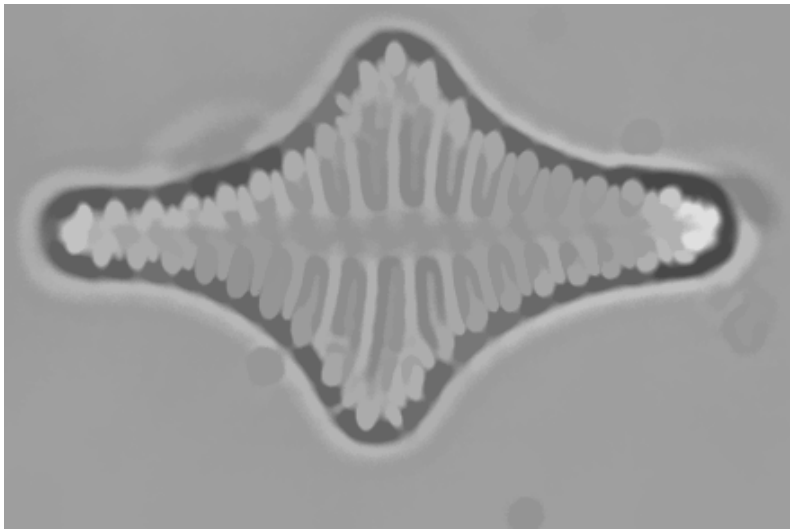
f



$\gamma^a_{256}(f)$



$\phi^a_{256}(f)$



$\psi^N_{256}(f)$

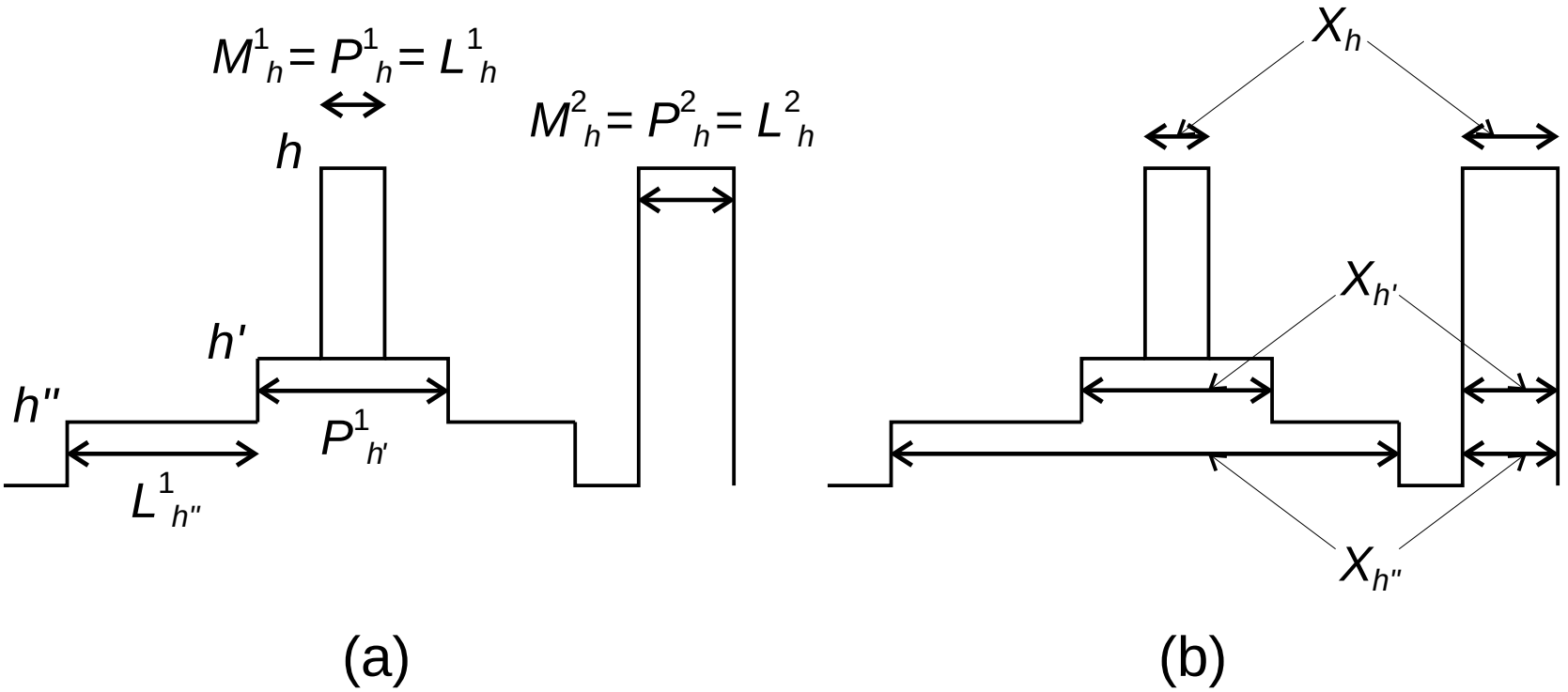
Definitions for Grey Scale

- A *level set* $\mathcal{L}_h$ of image f is defined as

$$\mathcal{L}_h(f) = \{x \in E | f(x) = h\} \tag{22}$$

- A *flat zone* or *level component* L_h at level h of a grey scale image f is a connected component of the level set $\mathcal{L}_h(f)$.
- *peak component* P_h at level h is a connected component of the thresholded set $X_h(f)$.
- A *regional maximum* M_h at level h is a level component no members of which have neighbors larger than h. A
- At each level h there may be several such components, which will be indexed as L_h^i, P_h^j and M_h^k, respectively.
- Any regional maximum M_h^k is also a peak component, but the reverse is not true.

Definitions for Grey Scale



One-dimensional example of level components, peak components and regional maxima.

Algorithms for the Grey-scale Case

- Naive computation of these filters in the grey-scale case can be done by threshold decomposition. This is SLOW!
- Three faster algorithms have been proposed
  - A priority-queue based approach (Vincent, 1993; Breen & Jones, 1996): low memory cost, time complexity $O(N^2 \log N)$.
  - A union-find approach (Meijster & Wilkinson 2002): low memory cost, time complexity $O(N \log N)$, fastest in practice, only for increasing filters.
  - The Max-tree based approach (Salembier *et al.*, 1998): high memory cost, time complexity $O(N)$, most flexible.
  - The Max-tree method was combined with the union-find method by Najman and Couprie (2003), and extended floating point (Geraud et al. 2007).
  - A variant of the Max-tree for second-generation connectivity was developed (Ouzounis and Wilkinson 2005, 2007)
  - A parallel variant has been developed recently (Wilkinson et al, 2008).

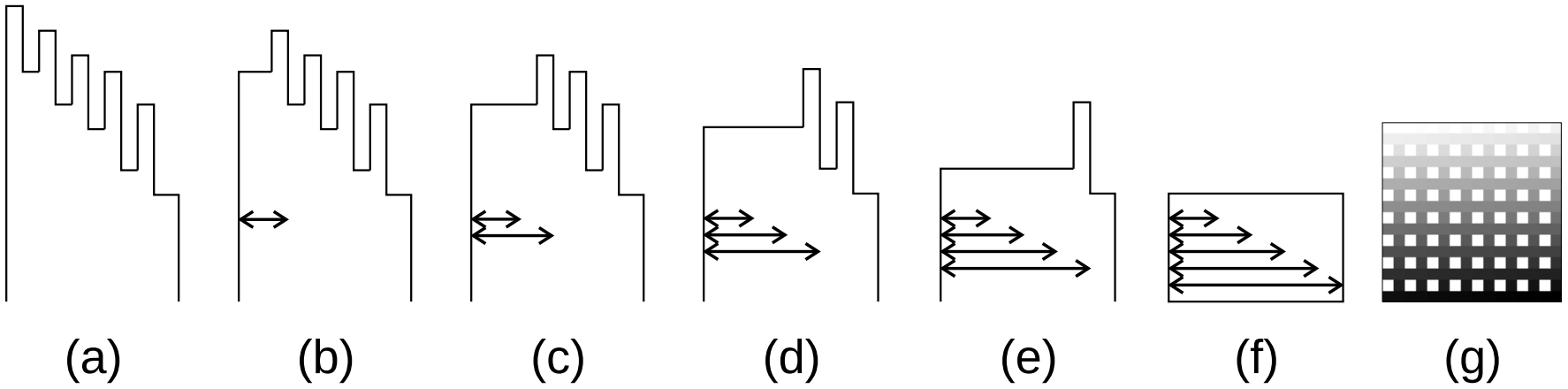
Priority-Queue Algorithm I

- Create a list of all regional maxima M_h^k.
- Select a seed pixel p_h^k from each maximum M_h^k.
- For each seed p_h^k do
  - push p_h^k in priority queue, with grey level as priority.
  - start flood-filling the peak component $P_{h'}^j$ around the regional maximum
  - Stop if either
    - the flooded area is equal to λ, or
    - a pixel is retrieved from the priority queue with grey value $h'' > h'$. In this case the region grown so far is not a peak component $P_{h'}^j$ *at level* h'.
  - Flood the region with grey value λ.
  - The algorithm terminates when all maxima have been processed.

The Core of the Priority-Queue Algorithm

```
/* List F contains the local maximum components */
while (F not empty) do
  {
     extract C from F;
     area =  A(C);
     curlevel = grey level of component;
     while (area < lambda)
       { n = neighbor of C with I[n]
             is maximum of all neighbors;
         if (I[n] > curlevel)
           break;
         else { add n to C;
                curlevel = I[n];
              }
       }
     for all p in C do
       {  I[p] = curlevel;
          L[p] = PROCESSED;
       }
  }
```

$O(N^2 \log N)$ Complexity



(a) Original 1-D signal on which the priority-queue algorithm shows its $O(N^2 \log N)$ behaviour.

(b)-(f) Processing sequence for each maximum, indicating pixels scanned before the next maximum is found.

(g) 2-D counterpart of (a).

Union-Find I

- We start from an observation that the partition of E induced by the connected components or level components of an image consist of disjoint sets.
- Tarjan's union-find algorithm for keeping track of disjoint sets can be used to implement merging in an efficient way.
- For each set (component) an arbitrary member is chosen as representative for that set.
- The algorithm uses rooted trees to represent sets, in which the root is chosen as the representative.
- Each non-root node in a tree points to its parent, while the root points to itself.
- Two objects `x` and `y` are members of the same set if and only if `x` and `y` are nodes of the same tree.
- This is equivalent to saying that they share the same root of the tree they are stored in.

Union-Find II

- There are four basic operations.
  - `MakeSet(x)`: Create a new singleton set $\{x\}$. This operation assumes that `x` is not a member of any other set.
  - `FindRoot(x)`: Return the root of the tree containing `x`.
  - `Union(x,y)`: Form the union of the two sets that contain `x` and `y`.
  - `Criterion(x,y)`: a symmetric criterion which determines whether `x` and `y` belong to the same set.

Union-Find III

- For flat zone labeling the algorithm becomes:

```
for all pixels p do
  { MakeSet(p);
    for all neighbors n<p do
      if ( I[n]==I[p] )
        Union( n, p );
  }
```

  Note that in this context the condition `n<p` means that `n` is a pixel which has been processed before `p`.

- This part finds the flat zones
- A second *resolving* phase is needed to assign labels.

Changes needed for Area Openings

- Pointers are replaced by integer indices refering to the location of the parent.
- Instead of letting the root point to itself, we set it to $-A(C)$, with $A(C)$ the area of the component gathered so far.
- We have to process the pixels in descending grey-scale order to be sure we process peak components from the top down.
- We do this by sorting the pixels first (counting sort $O(N)$), pixels of the same grey level are processed in lexicographic order.
- We link pixels p and q if:
  - $f(p) = f(q)$ or
  - ($f(p) > f(q)$ *and* `-parent[findroot[`p`]]`$< \lambda$) or
  - ($f(p) < f(q)$ *and* `-parent[findroot[`q`]]`$< \lambda$).
- We always choose the pixel with the *lowest* grey level as the root of a region.
- Resolving consists of assigning the root grey level to each pixel in a tree.

Basic Operations for Area Openings

```
void MakeSet ( int x )
{ parent[x] = -1;
}

int FindRoot ( int x )
{ if ( parent[x] >=0 )
    { parent[x] = FindRoot( parent[x] );
      return parent[x];
    }
  else return x;
}

boolean Criterion ( int x, int y )
{ return ( (I[x] == I[y]) ||
           ( -parent[x] < lambda ) ) ;
}
```

Basic Operations for Area Openings

```
void Union ( int n, int p )
{ int r=FindRoot(n);
  if ( r != p )
    { if ( Criterion(r, p) )
        { parent[p] = parent[p] + parent[r];
          parent[r] = p;
        }
      else
        parent[p] = -lambda;
    }
}
```

Area Opening by Union Find

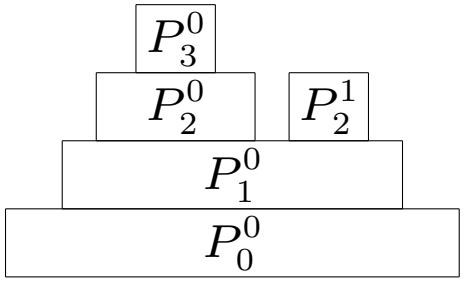
```
/* array S contains sorted pixel list */
for (p=0; p<Length(S); p++)
  {
    pix = S[p];  MakeSet(pix);
    for all neighbors nb of pix do
      if ((I[pix] < I[nb]) ||
          ((I[pix] == I[nb]) && (nb<pix)))
        Union(nb,pix);
  }
/* Resolving phase in reverse sort order */
for (p=Length(S)-1; p>=0; p--)
  {
    pix = S[p];
    if (parent[pix] >= 0)
        parent[pix] = parent[parent[pix]];
    else
        parent[pix] = I[pix];
  }
```

Tree Structures for Connected Filtering

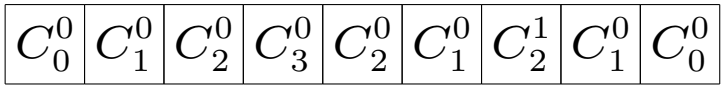
- Because peak components at different grey levels are nested within eachother, it is possible to represent the entire component structure as a tree.
- In *Max-trees* (Salembier et al., 1998) the nodes represent peak components.
- In *Min-trees* the nodes represent *valley components* (peak components of the inverted image).
- *Level-line trees* are built by computing a Min-tree and a Max-tree and merging these in such a way that the leaves of the tree are both minima and maxima in the image.
- Removing nodes in the Max-tree is leads to anti-extensive filtering
- Removing nodes in the Min-tree is leads to extensive filtering
- Removing nodes in the Level-line tree leads to auto-dual filtering.

Max-Tree representation

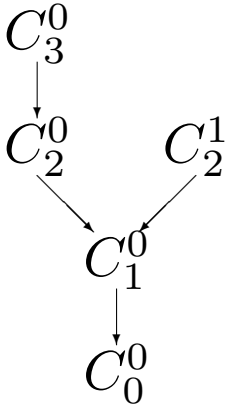
input signal



peak components

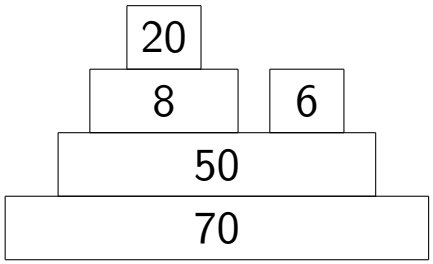


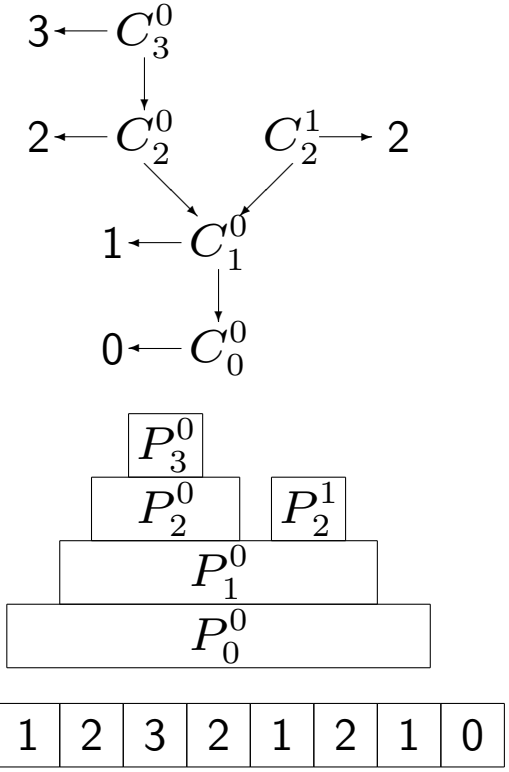
labelling

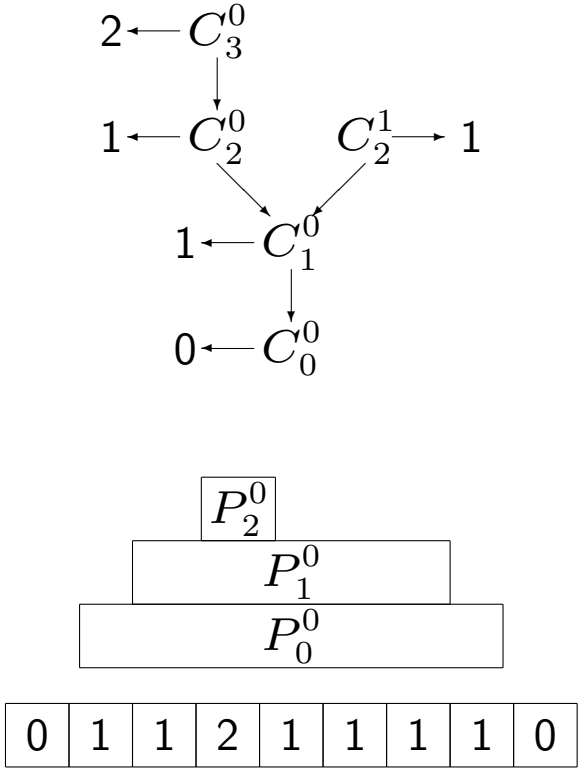


Max-Tree

Filtering



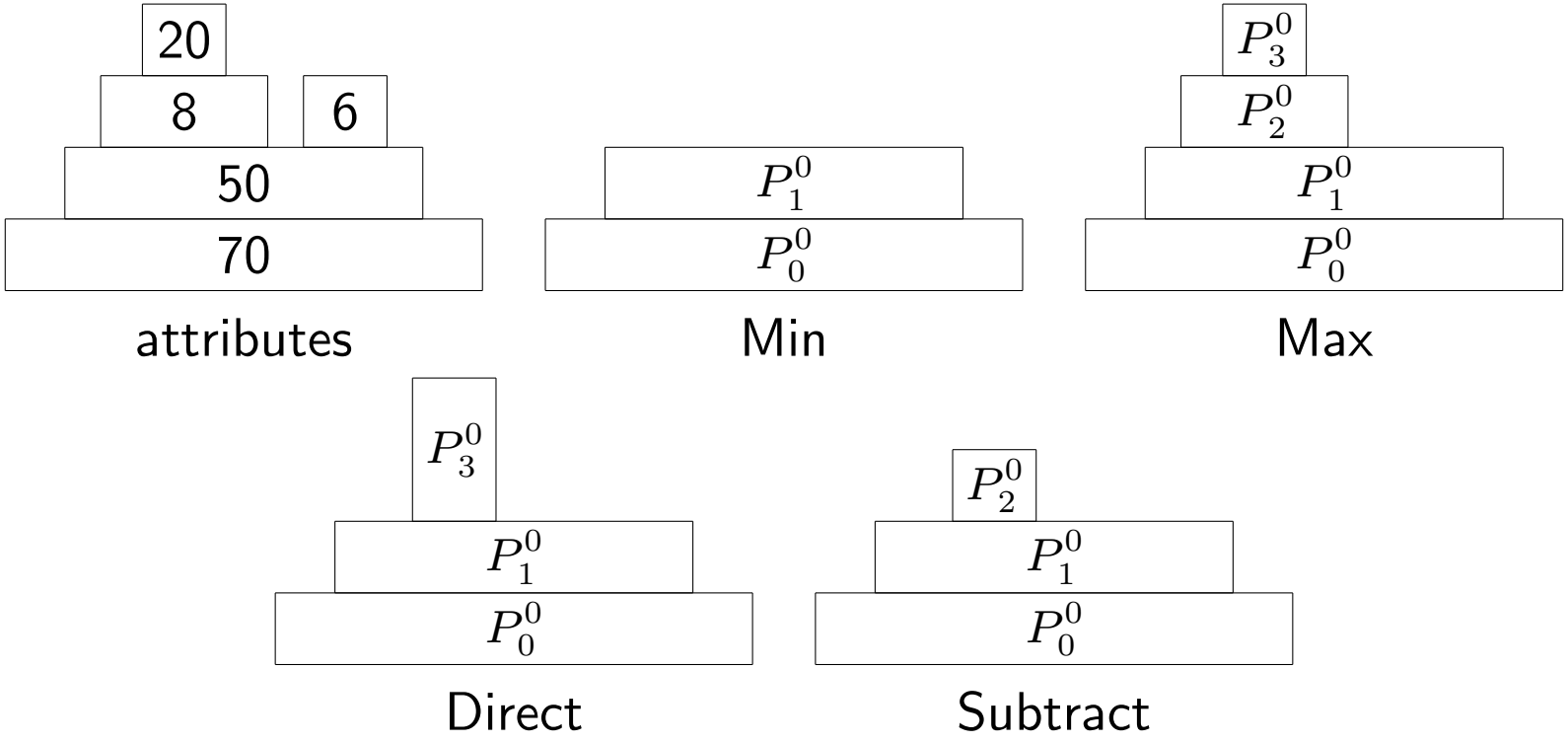




attribute values **original** **filtered** $(\lambda = 10)$

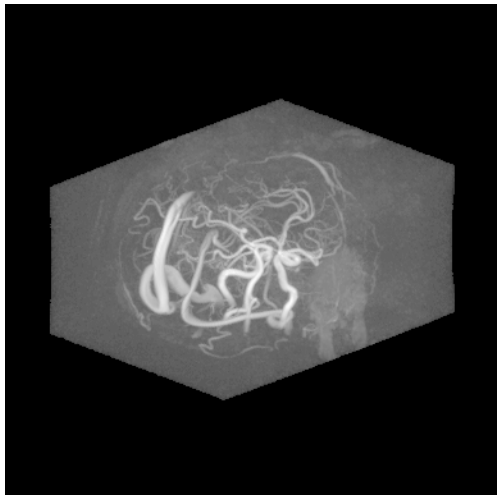
Filtering Rules

Different rules exist for removal of nodes:

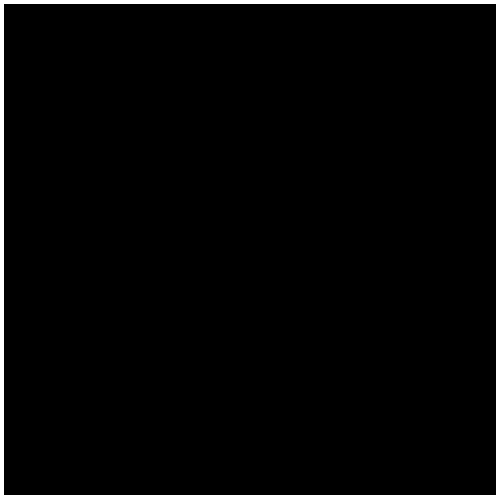


The first two are "pruning" rules, the second two "non-pruning". These different rules have an impact on the way "top-hat" equivalents of grey-scale shape filters work.

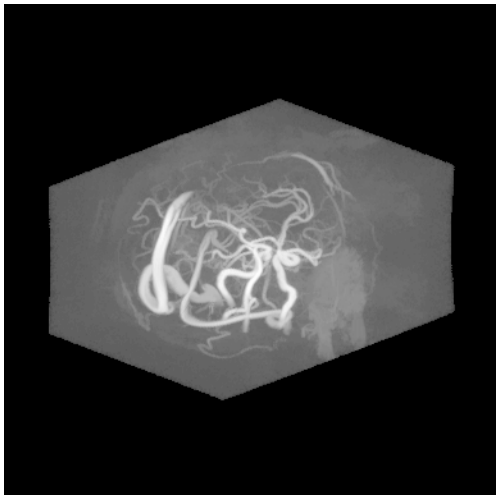
The Difference between Filtering Rules



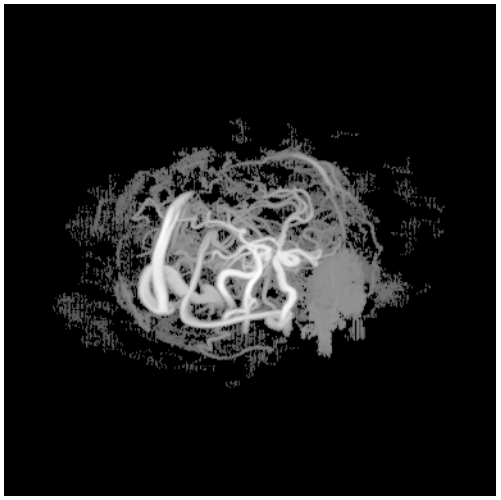
original



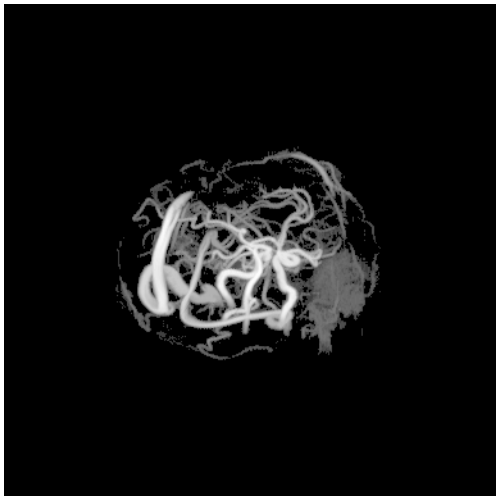
min



max



direct



subtractive

Invariances in Attribute Filters

- Very often in image analysis, we want our methods to be invariant to certain transforms.
- Most, if not all filters are shift invariant
- Rotation invariance can be obtained in structural filtering by:
  - Using a rotation invariant structuring element (SE), or
  - Using a non-rotation invariant SE at all possible rotations.
- In attribute filtering *invariance properties of the attribute carry over in the filter* if the connectivity is also invariant.
- Example: area is a rotation invariant attribute, and so is the area opening.
- Scale invariance is easily achieved in attribute filtering: use scale-invariant attributes: I/A^2.
- This leads to so-called *shape-filters*.

Why shape filters?

- Shape extraction is required whenever the objects of interest are characterized by shape, rather than scale.
- The common approach to this problem is by using multi-scale processing techniques.
- One example is finding elongated structures (vessels) is by using successive top-hat filters to obtain features of different width, followed by selection of sufficiently long features at each width scale by area openings.
- Multi-scale operators usually require multiple applications of filters to a single image.
- It may be more economical to design filters select for shape directly, in a single filter step.

Grey-Scale Image Decomposition by Shape

If we filter a grey-scale image f using shape criteria, we want the following properties to hold:

- *All* connected components of any threshold set of the filtered image $\phi_r^T(f)$ satisfy the shape criterion used.
- *None* of the connected components of any threshold set of the *difference* between the filtered image and original image $\phi_r^T(f) - f$ satisfy the shape criterion used

More formally we have

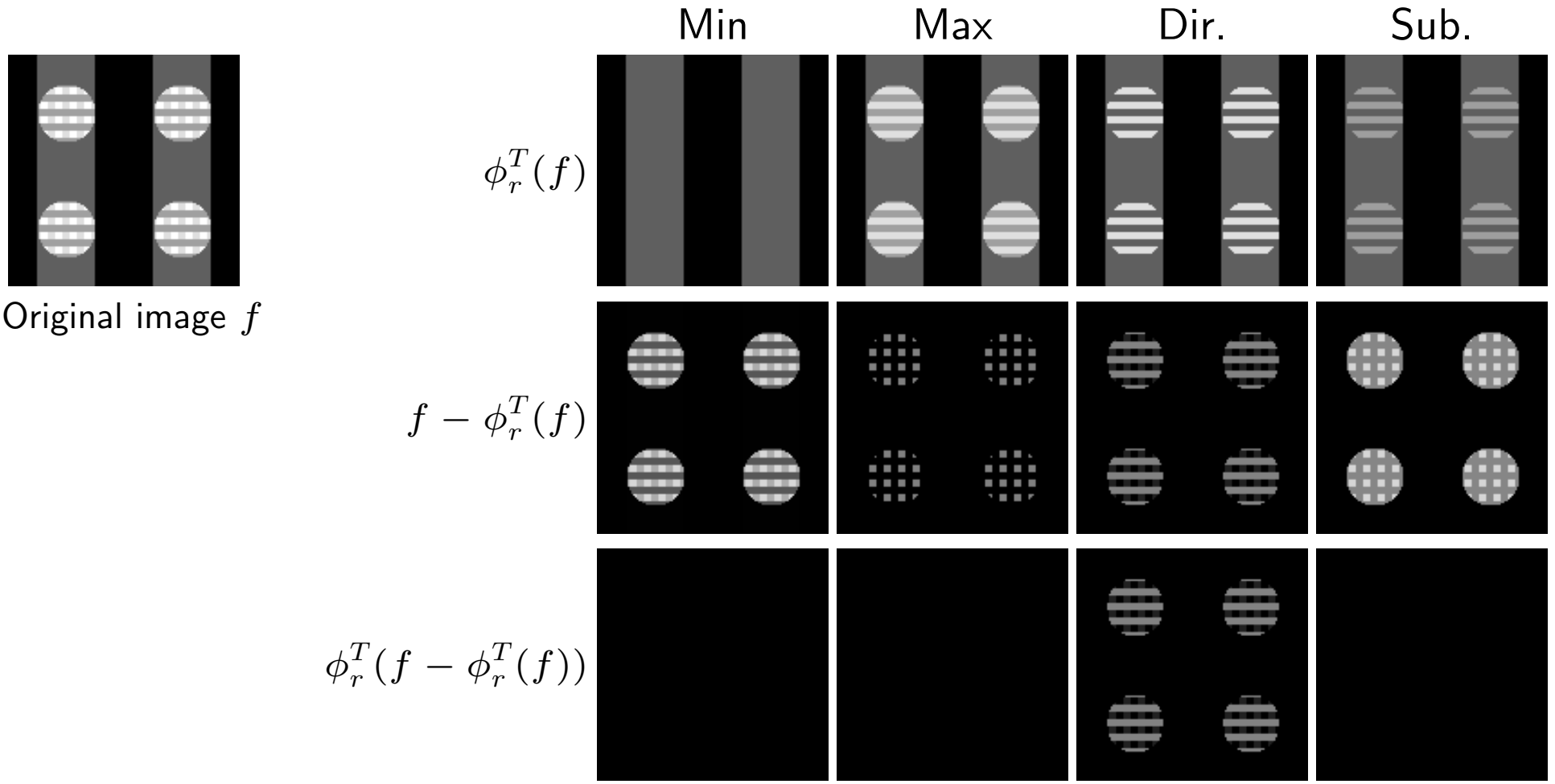
$$\Phi_r^T(X_h(\phi_r^T(f))) = X_h(\phi_r^T(f)) \tag{23}$$

and

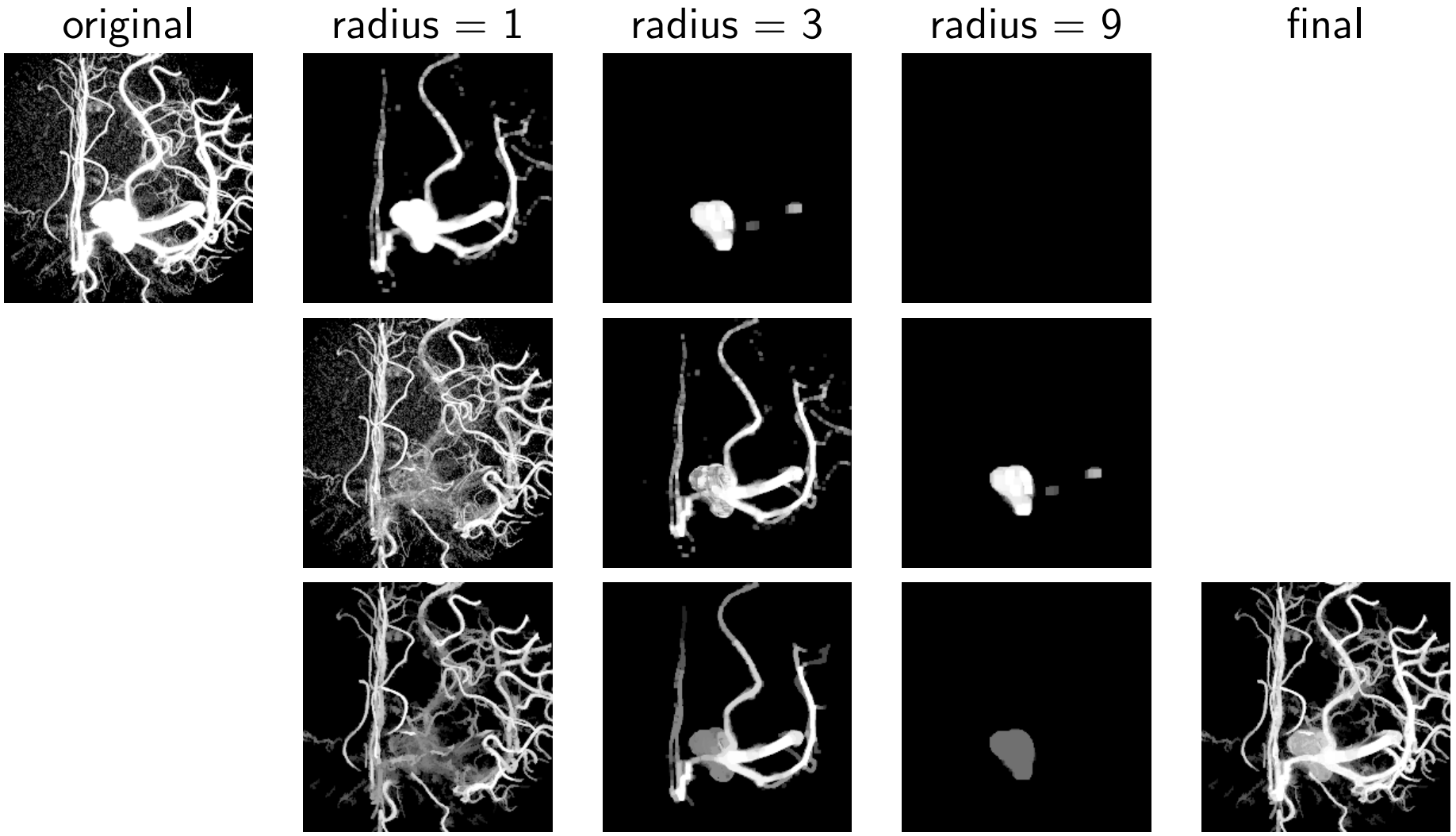
$$\Phi_r^T(X_h(f - \phi_r^T(f))) = \emptyset \tag{24}$$

for all h.

Grey-Scale Image Decomposition by Shape



Explicit Multiscale Approach



Shape filters: formal description

- Let us define a scaling X_λ of set X by a scalar factor $\lambda \in \mathbb{R}$ as

$$X_\lambda = \{x \in \mathbb{R}^n | \lambda^{-1} x \in X\}, \tag{25}$$

- An operator ϕ is said to be *scale invariant* if

$$\phi(X_\lambda) = (\phi(X))_\lambda \tag{26}$$

  for all $\lambda > 0$.

- If an operator is scale, rotation and translation invariant, we call it a *shape operator*.
- If it is also *idempotent* it is a *shape filter*.

An instance of a shape filter

- The binary connected opening Γ_x extracts the connected component to which x belongs, discarding all others.
- The trivial thinning Φ_T of a connected set C with criterion T is just the set C *if* C satisfies T, and is empty otherwise. Furthermore, $\Phi_T(\emptyset) = \emptyset$.
- The binary attribute thinning Φ^T of set X with criterion T is given by

$$\Phi^T(X) = \bigcup_{x \in X} \Phi_T(\Gamma_x(X)) \tag{27}$$

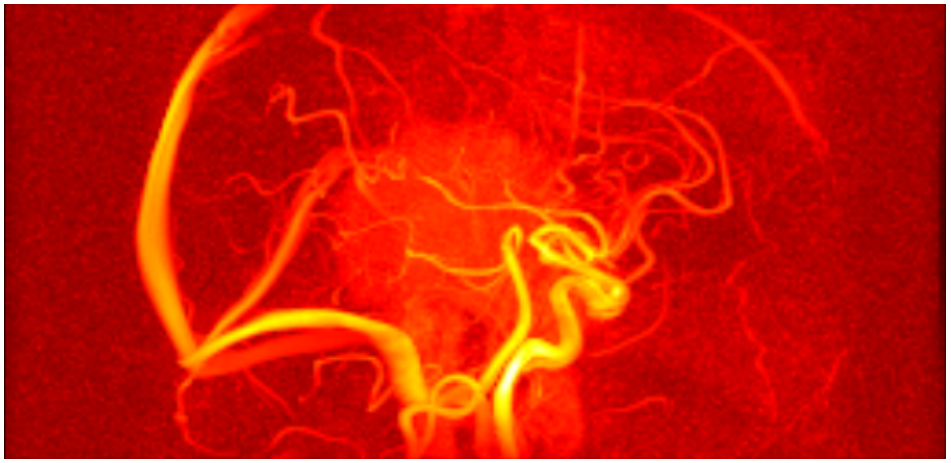
- If T is scale, rotation and translation invariant, Φ^T is a shape filter. An example would be:

$$T(C) = \left(\frac{I(C)}{A^2(C)} \geq \lambda \right). \tag{28}$$

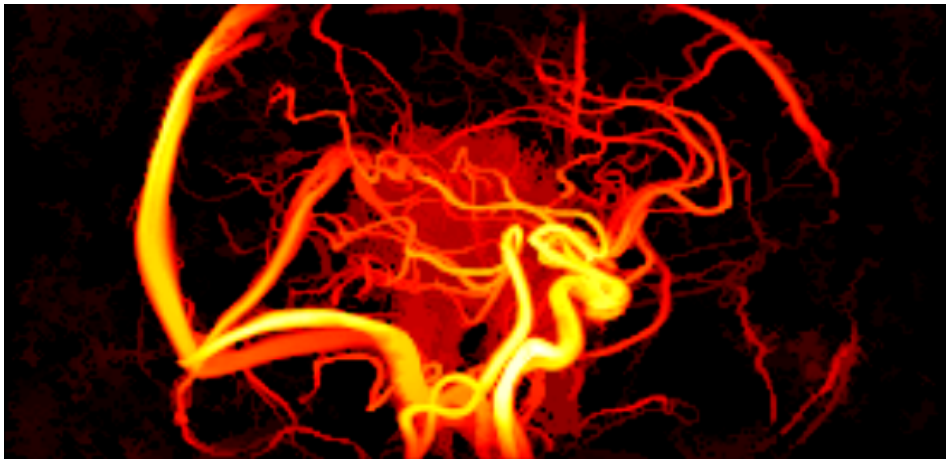
Vessel Enhancement

- In angiography it is often necessary to enhance curvilinear detail before segmentation.
- Standard multi-scale techniques require filtering at multiple scales *and* orientations.
- Shape filtering using 3D shape criteria can be used instead. Examples:
  - *non-compactness*: $I/V^{5/3} > \lambda$, in which I is the trace of moment-of-inertia tensor ($\propto$ covariance matrix of the coordinate distribution of the pixels) of the connected set, and V the volume.
  - *elongation* ϵ_1: ratio $|e_1|/|e_2|$ of the two largest eigenvalues of the moment-of-inertia tensor
  - *flatness* f_1: ratio $|e_2|/|e_3|$ of the two smallest eigenvalues.
  - *elongation* ϵ_2: ratio $|e_1|/\sqrt{|e_2 e_3|}$
  - *flatness* f_2: ratio $\sqrt{|e_1 e_2|}/|e_3|$.
- The result can be computed in under 12 s on a Pentium 4 at 1.9 GHz for a 256^3 volume.

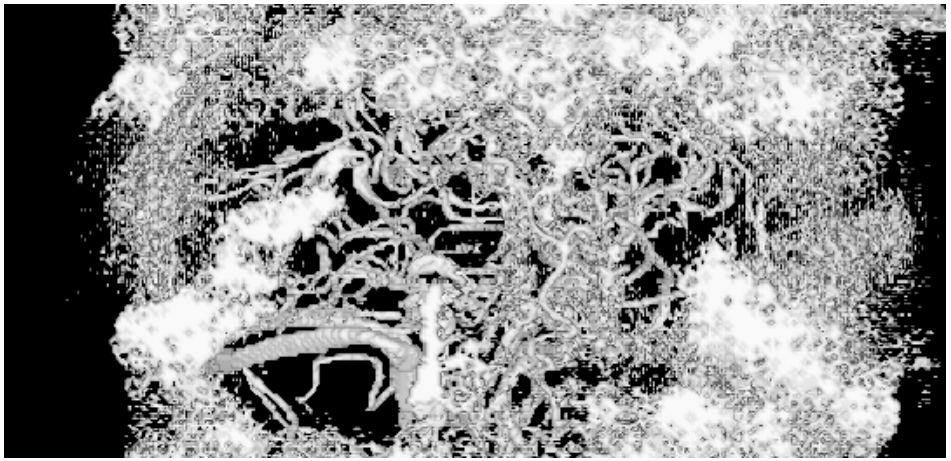
Vessel Enhancement II



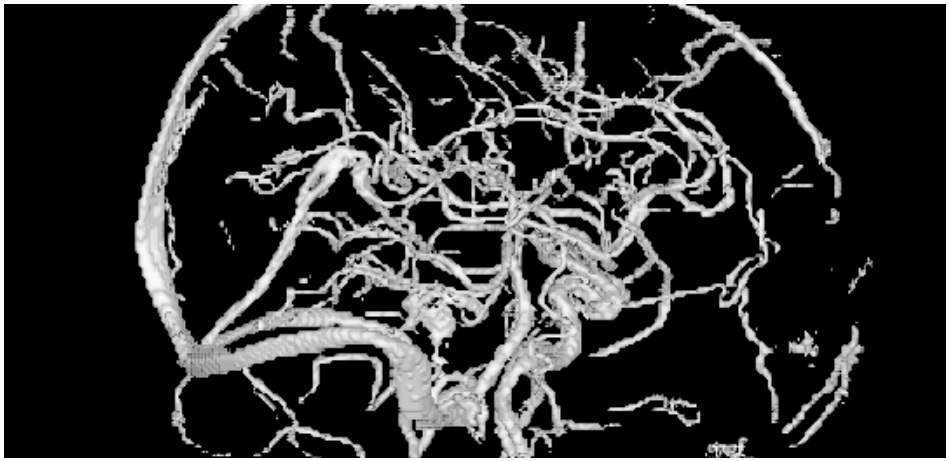
angiogram



filtered $\lambda = 2.0$

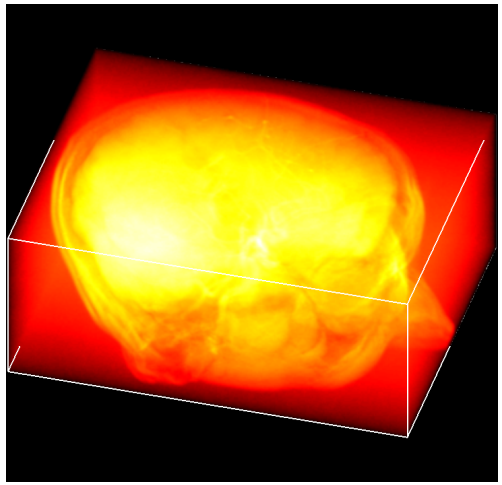


segmentation of original

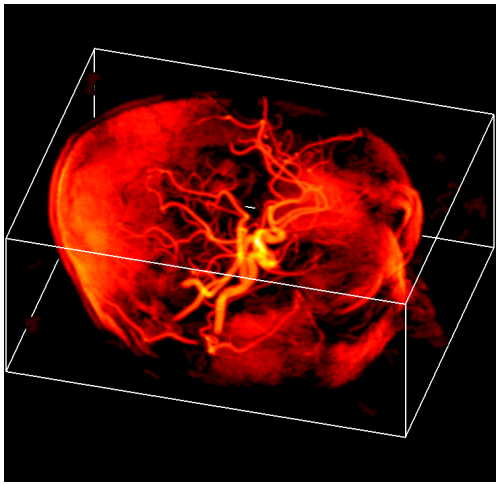


segmentation of filtered set

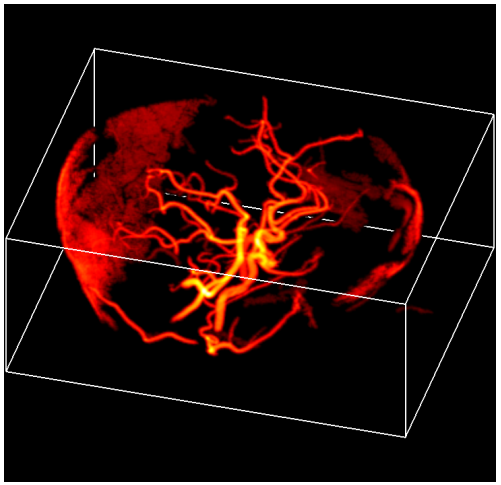
Problem: Time of Flight MRA



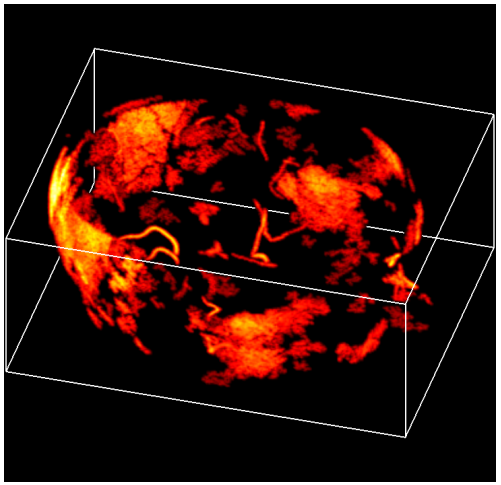
original



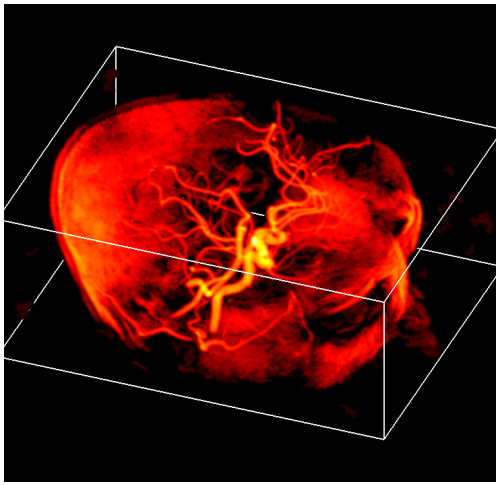
$I/V^{5/3} \geq 2.0$



$I/V^{5/3} \geq 4.8$



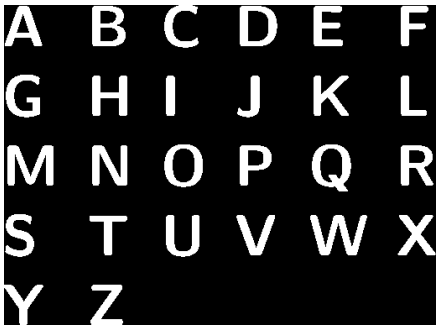
$f_1 \geq 19.0$



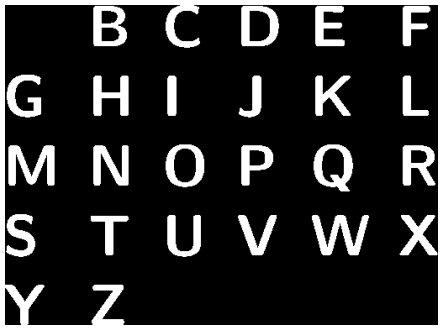
$I/V^{5/3} \geq 2.0$ and $f_1 < 19.0$

Vector-attribute filters

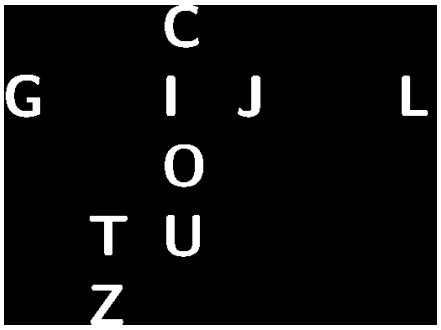
- Aim: Removing objects that are similar enough to a given shape.
- Example: removing objects that are similar enough (ϵ) to the reference shape (letter A).



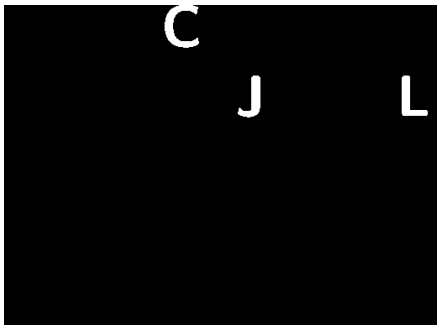
Original image X



$\epsilon = 0.01$



$\epsilon = 0.10$



$\epsilon = 0.15$

- A value of $\epsilon = 0$ means only those shapes are removed that are exactly the same as the reference shape.
- To gain more descriptive power we may use more than one attribute per node.

Binary Vector-Attribute Thinning

- A multi-variate attribute thinning $\Phi^{\{T_i\}}(X)$ with scalar attributes $\{\tau_i\}$ and their corresponding criteria $\{T_i\}$, with $1 \leq i \leq N$, preserves a component C if $\exists i : T_i$, $T_i = \tau_i(C) \geq r_i$:

$$\Phi^{\{T_i\}}(X) = \bigcup_{i=1}^{N} \Phi^{T_i}(X). \tag{29}$$

- An alternative is the *vector-attribute thinning*, in which C is preserved if $\vec{\tau}(C) \in \mathbb{R}^D$ satisfies criterion

$$T^{\vec{\tau}}_{\vec{r},\epsilon}(C) = d(\vec{\tau}(C), \vec{r}) \geq \epsilon \tag{30}$$

  in which dissimilarity measure $d : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ quantifies the difference between $\vec{\tau}(C)$ and $\vec{r}$.

- A binary vector-attribute thinning $\Phi^{\vec{\tau}}_{\vec{r},\epsilon}(X)$, with D-dimensional vectors, removes the connected components of a binary image X whose vector-attributes differ less than ϵ from a reference vector $\vec{r} \in \mathbb{R}^D$.

Vector Attribute Thinning

Definition 1. *The vector-attribute thinning $\Phi^{\vec{\tau}}_{\vec{r},\epsilon}$ of X with respect to a reference vector $\vec{r}$ and using vector-attribute $\vec{\tau}$ and scalar value ϵ is given by*

$$\Phi^{\vec{\tau}}_{\vec{r},\epsilon}(X) = \{x \in X |\ T^{\vec{\tau}}_{\vec{r},\epsilon}(\Gamma_x(X))\}. \tag{31}$$

Possible choices for d:

- Euclidean distance $d(\vec{u},\vec{v}) = ||\vec{v} - \vec{u}||$.
- Manhattan distance $d(\vec{u},\vec{v}) = \sum |v_i - u_i|$
- Any dissimilarity measure can be used (such as Mahalanobis distance).
- Since the triangle inequality $d(a,c) \leq d(a,b) + d(b,c)$ is not required, d need not be a distance.

Thinning with Respect to a Shape (Family)

- To select the appropriate vector $\vec{r}$ we can provide a shape in a binary image and compute its vector attributes.

  Definition 2. *The vector-attribute thinning* $\Phi^{\vec{\tau}}_{S,\epsilon}$ *of* X *with respect to a reference* shape S *and using vector-attribute* $\vec{\tau}$ *and scalar value* ϵ *is given by*

$$\Phi^{\vec{\tau}}_{S,\epsilon}(X) = \Phi^{\vec{\tau}}_{\vec{\tau}(S),\epsilon}(X) \tag{32}$$

- More robustness can be obtained using a series of example shapes in a shape family $F = \{S_1, S_2, \ldots, S_n\}$:

  Definition 3. *The vector-attribute thinning* $\Phi^{\vec{\tau}}_{F,\epsilon}$ *of* X *with respect to a reference* shape family F *and using vector-attribute* $\vec{\tau}$ *and scalar value* ϵ *is given by*

$$\Phi^{\vec{\tau}}_{F,\epsilon}(X) = \bigcap_{S \in F} \Phi^{\vec{\tau}}_{S,\epsilon}(X) \tag{33}$$

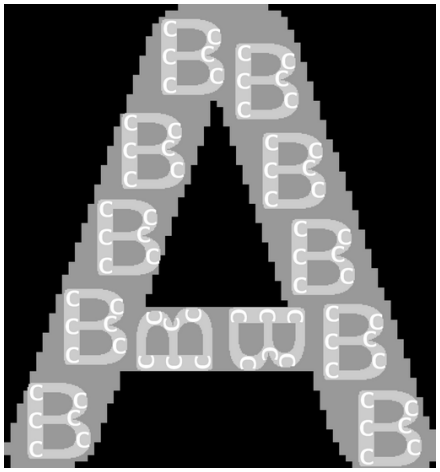
- This removes objects if they are similar enough to any of the example shapes

Gray-scale vector-attribute thinning

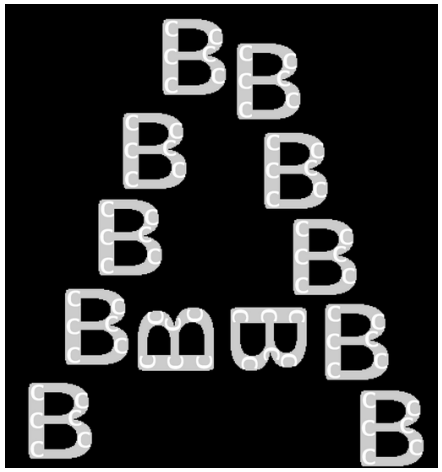
Extension to gray-scale using threshold decomposition:

$$\phi^{\vec{\tau}}_{\vec{r},\epsilon}(f) = \sup\{h|\ T^{\vec{\tau}}_{\vec{r},\epsilon}(\Gamma_x(X_h(f)))\}, \tag{34}$$

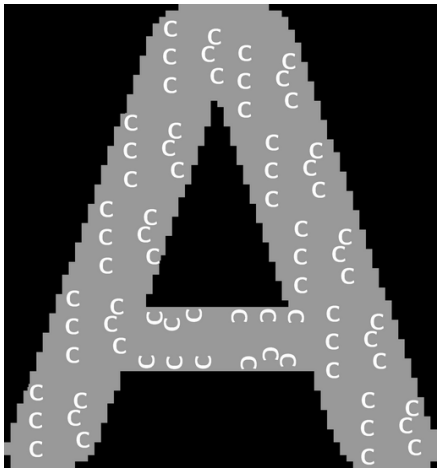
where threshold set $X_h(f)$ is defined as: $X_h(f) = \{x \in \mathbf{M} | f(x) \geq h\}$. Example: removing letters from image f consisting of nested versions of the letters A, B, and C.



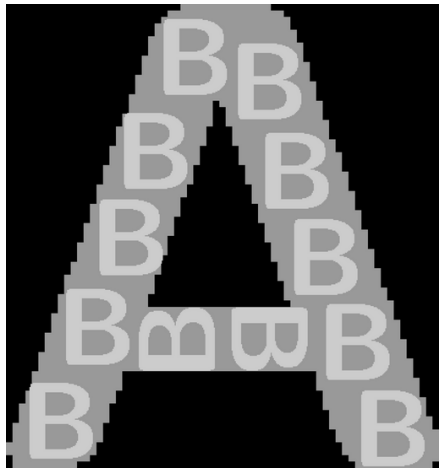
f



$\phi^{\vec{\tau}}_{S_A,\epsilon}(f)$



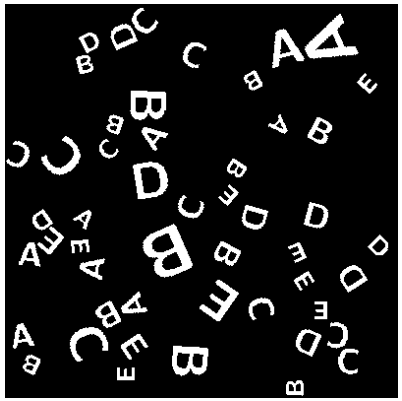
$\phi^{\vec{\tau}}_{S_B,\epsilon}(f)$



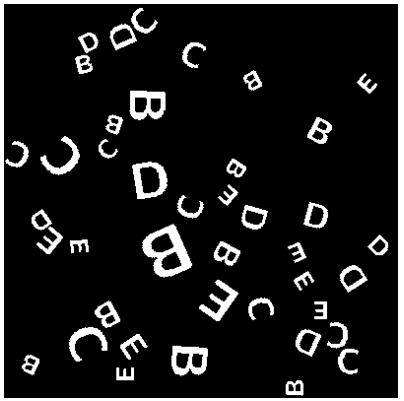
$\phi^{\vec{\tau}}_{S_C,\epsilon}(f)$

Filtering using Hu's Moment Invariants

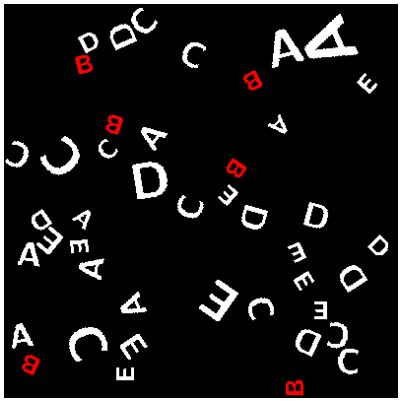
Using vector-attribute thinning with Hu's set of 7 moment invariants as vector-attribute to remove from image X the letters A, B, and C respectively.



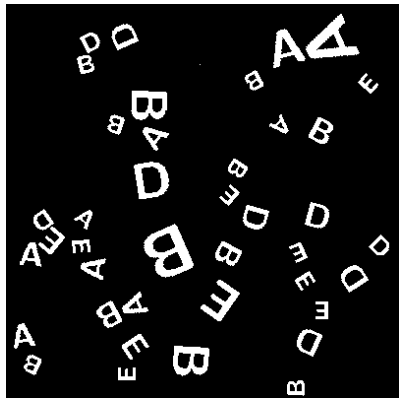
X



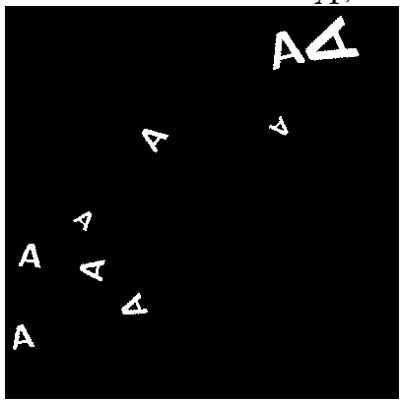
$\Phi^{\vec{\tau}}_{S_A,0.010}(X)$



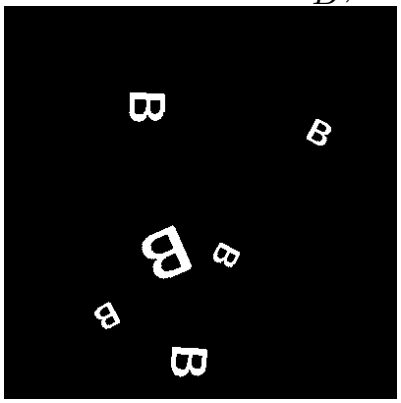
$\Phi^{\vec{\tau}}_{S_B,0.013}(X)$



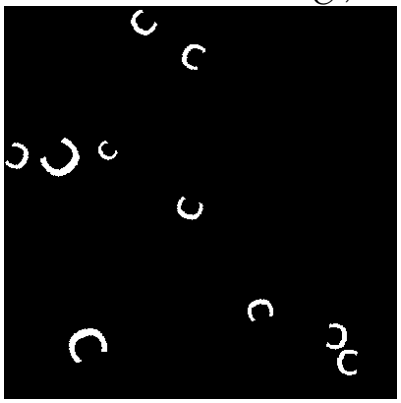
$\Phi^{\vec{\tau}}_{S_C,0.010}(X)$



$X - \Phi^{\vec{\tau}}_{S_A,\epsilon}(X)$



$X - \Phi^{\vec{\tau}}_{S_B,\epsilon}(X)$



$X - \Phi^{\vec{\tau}}_{S_C,\epsilon}(X)$

Traffic-Sign Recognition

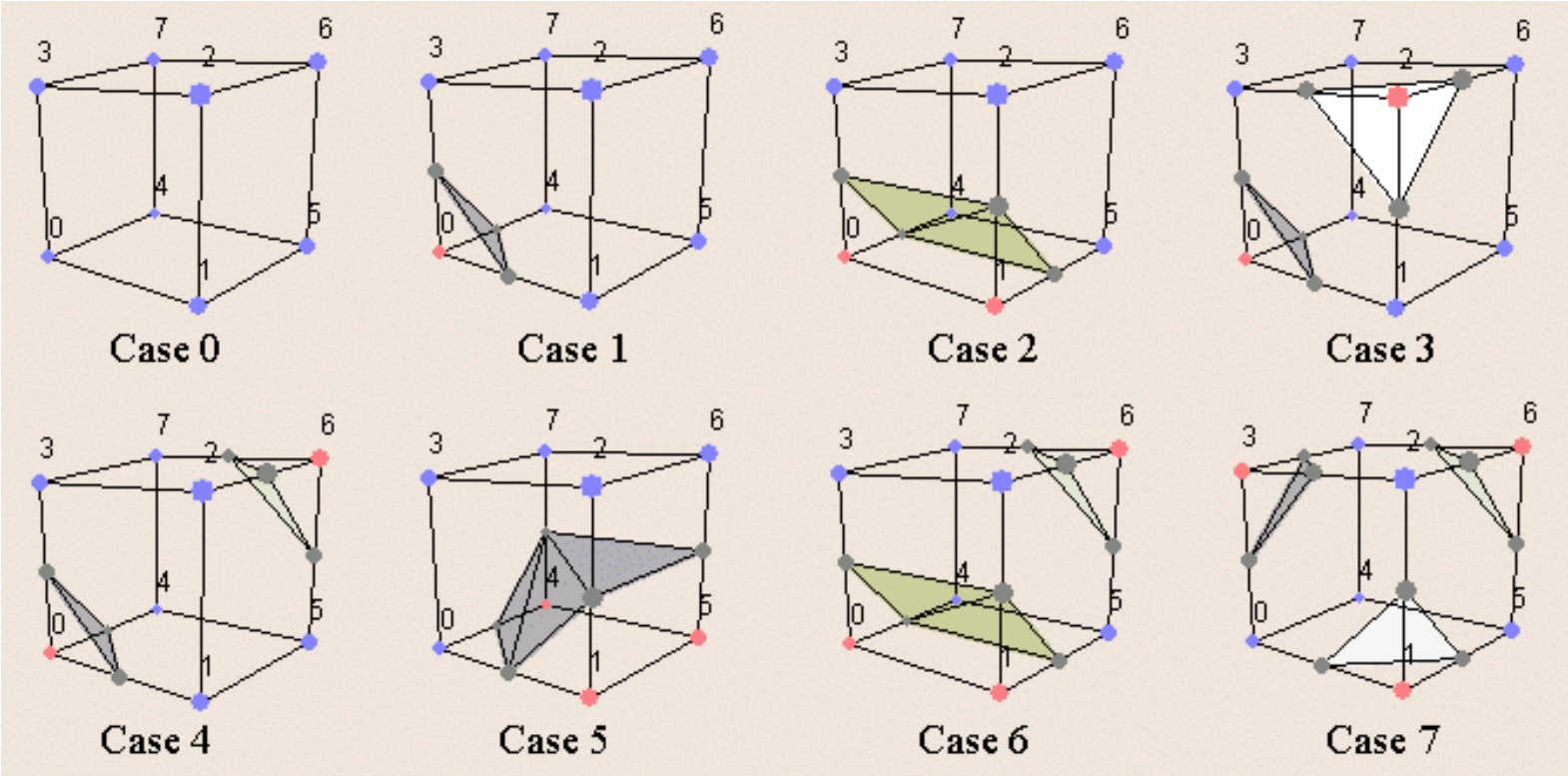
- Using scaling and rotation invariant vector-attribute filters it is possible to detect e.g. traffic signs in natural scenes.
- Using a single filter it is possible to detect multiple targets.

Visualization using Max-trees

- Visualization of 3-D data sets can be done in a variety of ways.
- One class of rendering methods first extracts some interim representation in terms of graphical primitives from the data
- Once extracted, this representation can be rendered rapidly using standard graphical systems, e.g. through OpenGL.
- The most common example is iso-surface rendering
- Alternatively, we can use *direct volume rendering* in which the 3-D data are projected directly onto the view plane in some way.
- Examples are Maximum Intensity Projection (MIP) and X-ray rendering
- The following techniques can be handled efficiently through Max-trees:
  - Iso-surface rendering
  - Splatting (yielding either MIP or X-ray rendering)
  - Texture-based Volume Rendering (as above and more advanced transfer functions).

Isosurface

- 3-D surface representing locations of constant scalar value within volume
- Standard method: marching cubes



Augmenting the Max-Tree

- **Definition:** the *root path* of node C_h^k contains all nodes encountered on the descent from C_h^k to the root
- Consider 26-connected neighborhood, then:
  1. all 8 corner voxels of a cell are part of the same root path
  2. filtering does not change the grey-level ordering of nodes along a root path
- Therefore:
  - one node $C_{h_1}^k$ defines a cell's minimum
  - another node $C_{h_2}^k$ defines a cell's maximum
- **Important:** after filtering, the same nodes still define the cell's minimum and maximum

Augmented Max-Tree

0	2	1	0
0	3	1	0
0	2	1	0
0	2	1	0

original image

C_0^0	C_2^0	C_1^0	C_0^0
C_0^0	C_3^0	C_1^0	C_0^0
C_0^0	C_2^0	C_1^0	C_0^0
C_0^0	C_2^0	C_1^0	C_0^0

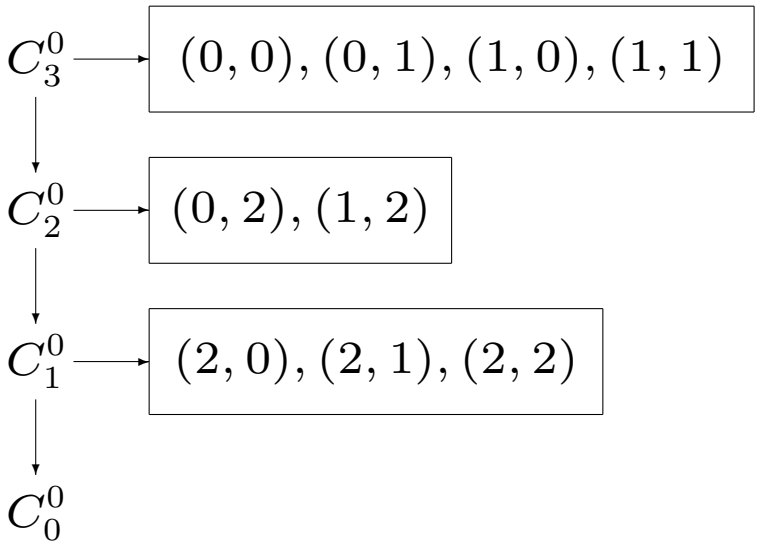
label image

C_0^0	C_1^0	C_0^0	
C_0^0	C_1^0	C_0^0	
C_0^0	C_1^0	C_0^0	

$V_{\min}$

C_3^0	C_3^0	C_1^0	
C_3^0	C_3^0	C_1^0	
C_2^0	C_2^0	C_1^0	

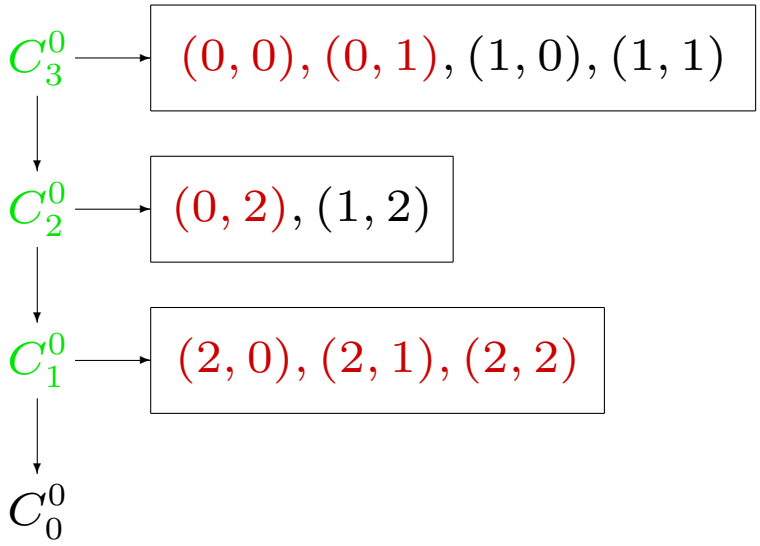
$V_{\max}$



Augmented Max-tree

Pseudo-code

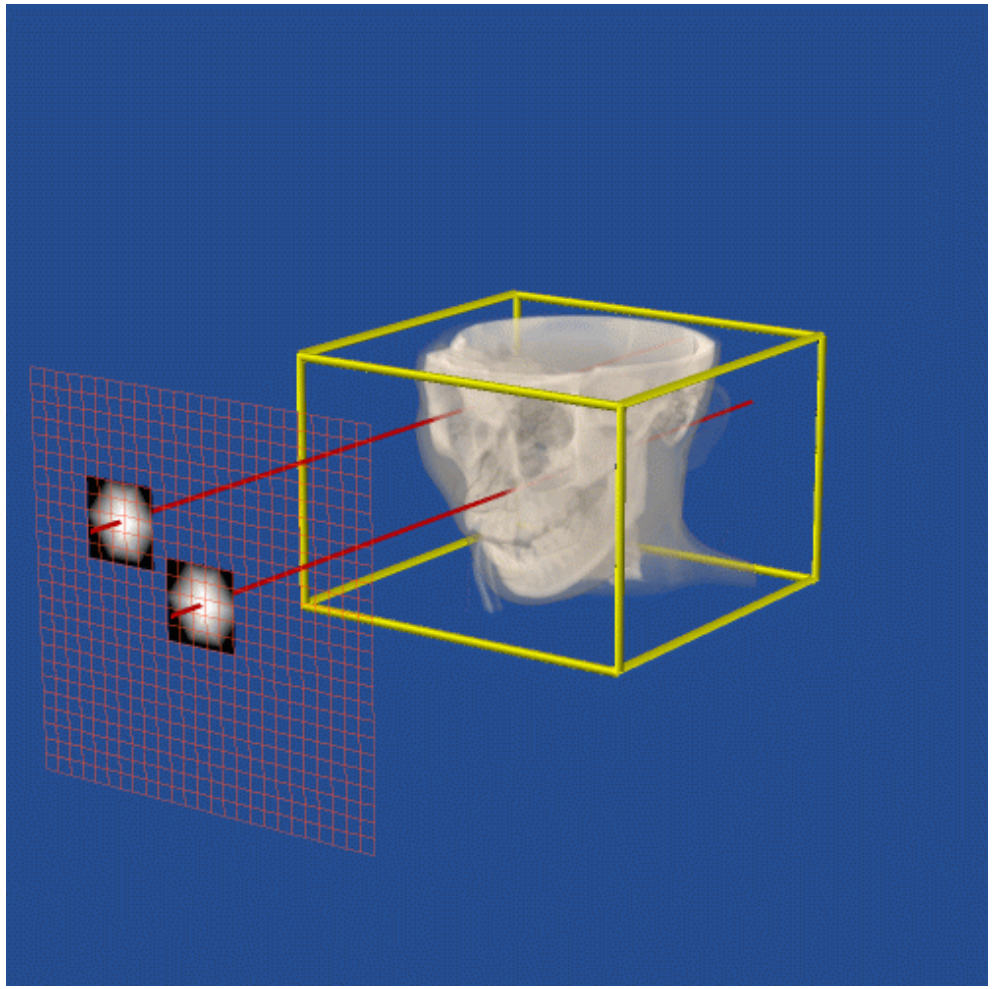
for all nodes p **do**
  p.processed $\leftarrow$ false
end for
root.processed $\leftarrow$ true
for all leaves q **do**
  p $\leftarrow$ q
  while (not p.processed) and $(g(\mathsf{p}) \geq t)$ **do**
    i $\leftarrow$ 0
    while (i<p.numEdges) and $(g(V_{\mathsf{min}}(c_i^p)) \leq t)$ **do**
      mark c_i^p as active
      i $\leftarrow$ i + 1
    end while
    p.processed $\leftarrow$ true
    p $\leftarrow$ p.parent
  end while
end for



Visited nodes and **cells** for $t = 0.5$.

Isosurface result

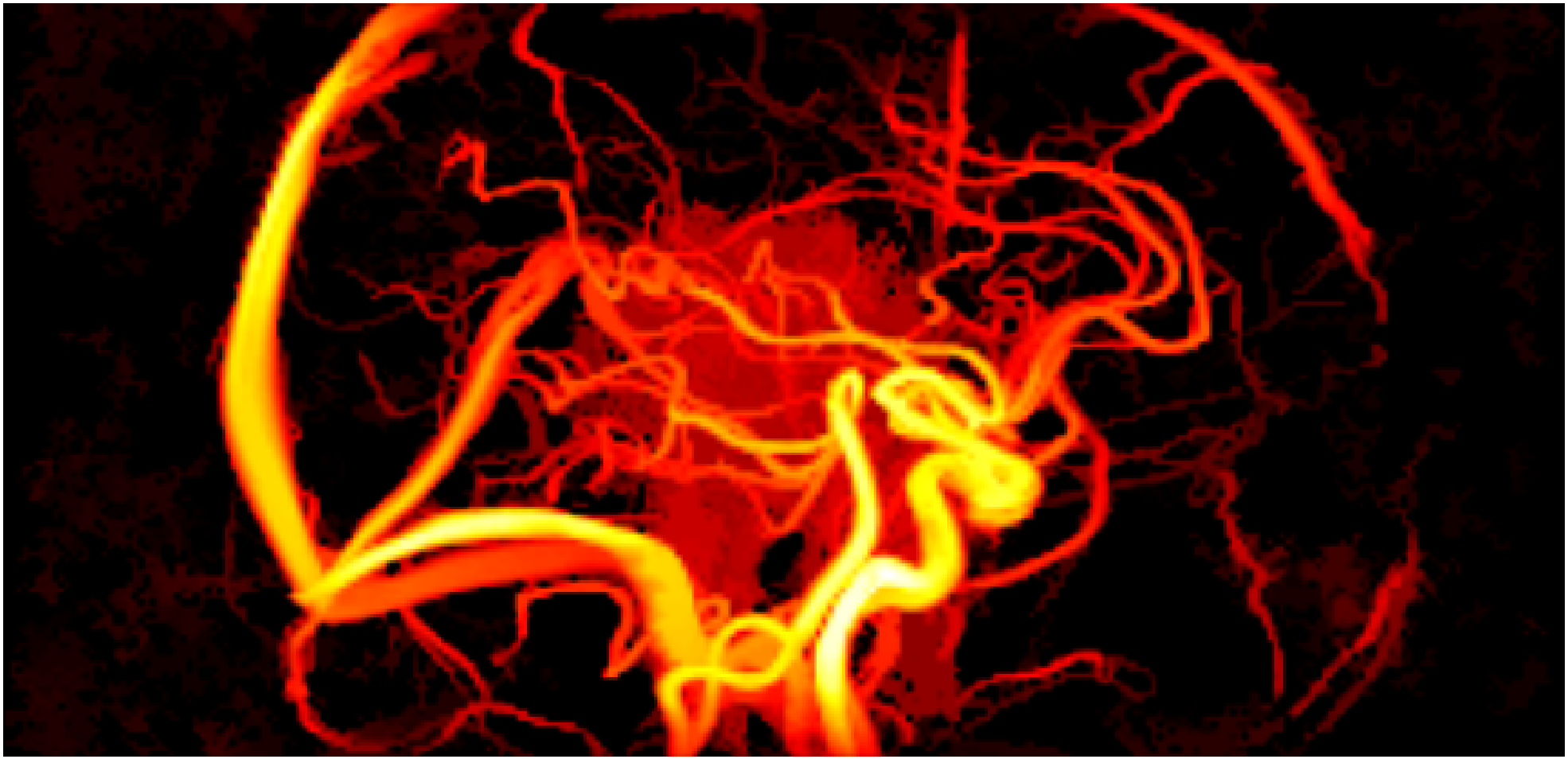
Splatting: the principle



Splatting from a Max-tree

- In standard X-ray and maximum-intensity projection, visiting zero grey-level voxels wastes time.
- Rather than filtering and rebuilding a volume, extract the non-zero voxels from the Max-tree, and splat these.
- In the orginal Max-tree representation, it was easy to find which node a voxel belongs to, but not the reverse.
- Adding a voxel-list to each node circumvents this problem
- simply scan the Max-tree from the leaves downwards, and splat each voxel of each non-zero node.

Splatting result



Texture-Based Volume Rendering

- Instead of using the CPU to perform most of the work, we can use the GPU to perform the rendering through texture-based volume rendering
- In the standard approach (STBV), the complete volume is loaded into graphics memory as a 3-D texture
- It can then be rendered using any *transfer function*, which dictates how grey levels are represented (e.g. through a colour/tranparency LUT)
- *Blending methods* also determine the rendering method (MIP, X-Ray,etc).
- Advantages of the method are:
  - Speed when changing viewpoint
  - Versatility
- Drawbacks are
  - Requires large graphics memory
  - Slow when doing λ-browsing

Indirect Texture-Based Rendering

- We can increase the speed of λ-browsing by using *indirect* texture-based volume rendering.
- In this case the texture data are first sent through a look-up table (LUT), and the LUT value is used to determine the transfer function result.
- We can now send a *label* volume to the texture buffer once, as a 3-D texture T_v.
- Each voxel in T_v contains a label corresponding to its Max-tree node C_h^k.
- We use a second (1-D) texture T to encode the grey levels for each node C_h^k.
- Whenever λ is changed, we update T by copying the new grey levels of each node (stored in an array $A[i]$ for convenience), and send *only* T to the graphics board.
- We can then draw by indirect texture-based rendering, purely on the GPU!

Algorithm

Update()

{$g(p)$ denotes current grey value of node p}
for $i \leftarrow 0$ to length(A) **do**
  $T[i] \leftarrow g(A[i])$
end for
Transfer T to graphics hardware

Draw()

for all slice planes s **do**
  for all fragments f in s **do**
    $i \leftarrow$ sample T_v in point f {Fetch node index}
    $g \leftarrow T[i]$ {Fetch current grey value}
    f.color $\leftarrow$ TransferFunction(g)
  end for
  Blend s with frame buffer
end for

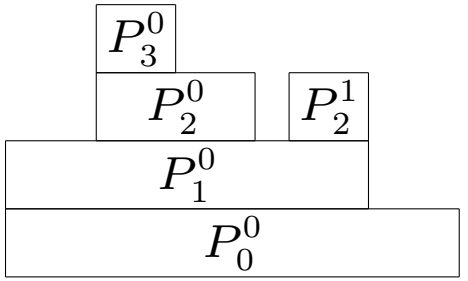
Parallel Computation

- Parallel computation is desirable because the volume data sets are often hughe ($512^3 = 256$ MB for short integer data).
- Parallel computation of connected filters is hard, because
  - Connected filters are not local
  - Connected filters are not separable
- We have developed a parallel algorithm for extensive and anti-extensive attribute filters by:
  - dividing the image (or volume) into strips
  - computing a Max-tree for each strip
  - merging the local Max-trees into a single tree
  - performing the filtering strip-wise
- To do this efficiently, we need to merge the Max-tree and union-find approaches.
- The big problem is keeping the attributes correct

Including Union-find

1	1	3	2	2	1	2	0	0

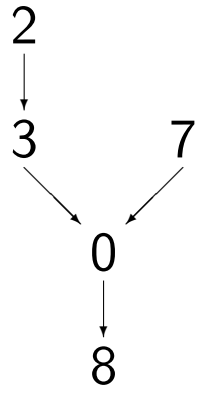
input signal



peak components

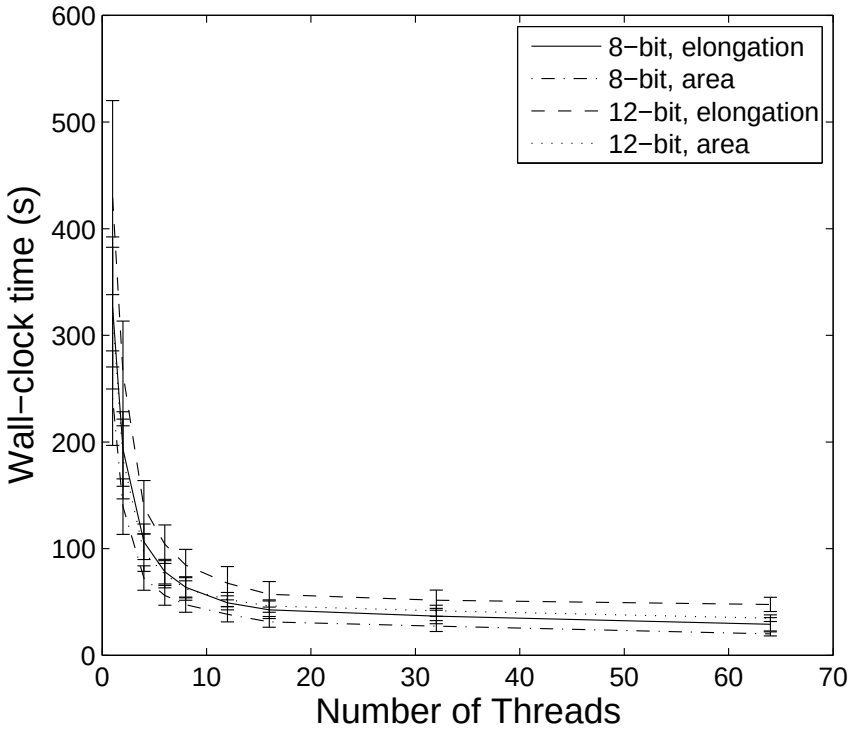
0	0	2	3	3	0	7	8	8

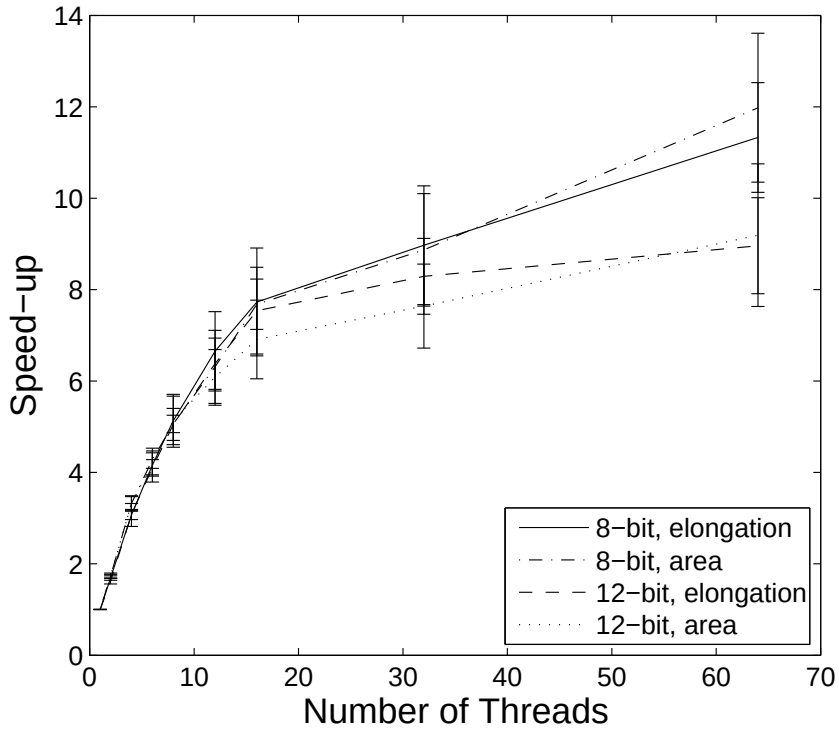
labelling



Max-Tree

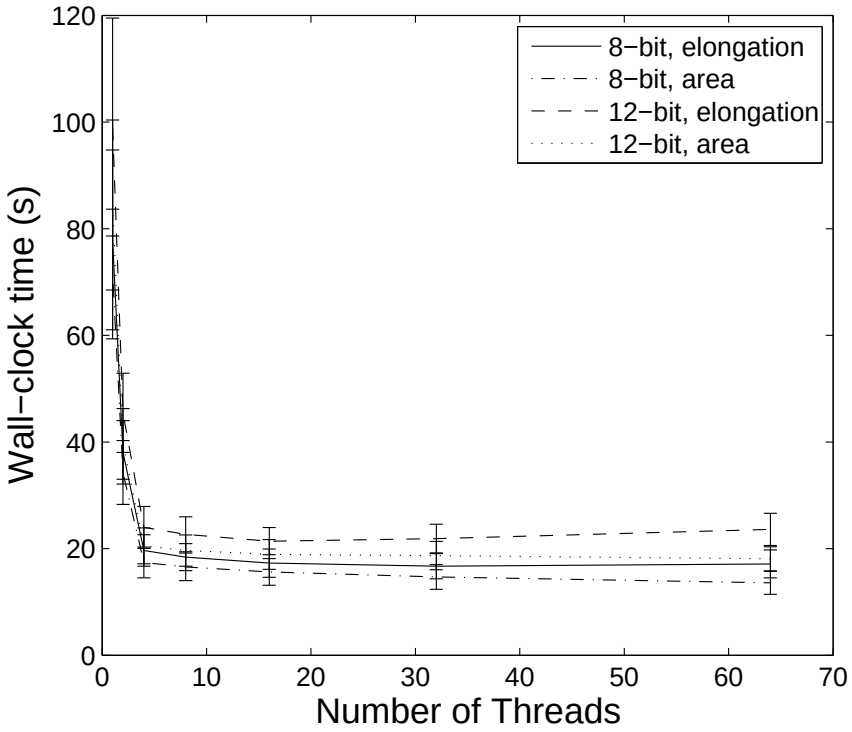
Speed-up

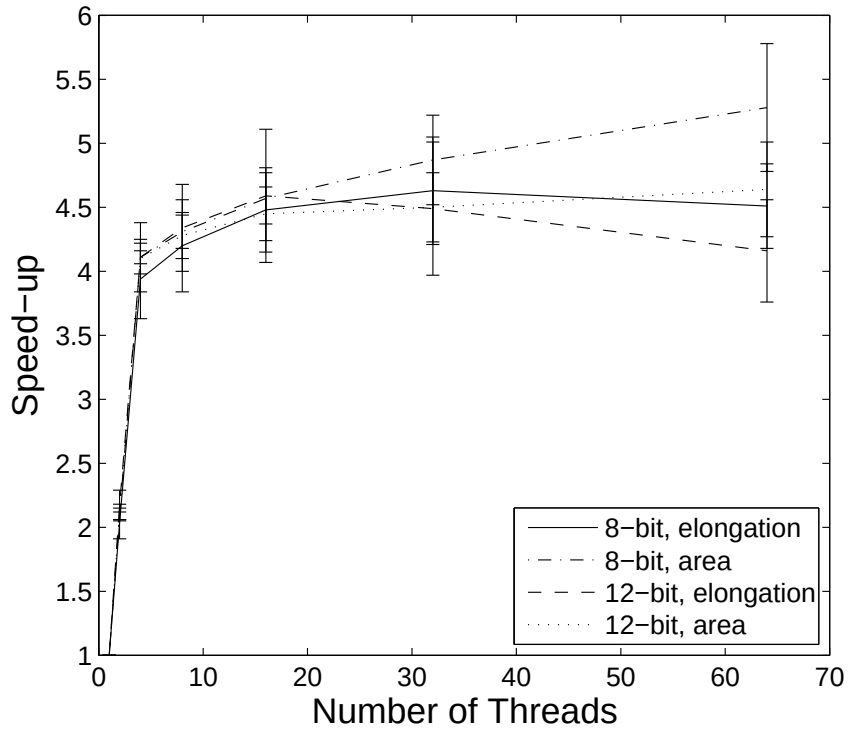




Timings were performed on the 16 CPU Onyx 3400 of the Centre for High Performance Computing & Visualisation of the RuG.

Speed-up II





Timings on a 2 socket dual-core Opteron-based machine.

Other work

- Morphological connected hat scale-spaces based on Max-trees have been constructed for contour and texture analysis.
- The C-trees for multi-scale connectivity analysis of binary images as suggested by Tzafestas & Maragos (2003) can be implemented rapidly as Max-trees of opening transforms.
- Derived connectivities (i.e. using openings or closings) can be incorporated into the Max-tree by constructing the tree not from one, but from two images. The second image encodes the altered connectivity.
- Extending the attributes for shape filtering.
- Making shape filters trainable by examples.

Questions

?