Towards Reliable Low-Level Image Analysis

Vorlesung Bildinformationssysteme, Teil 8
Sommersemester 2005
Ullrich Köthe, FB Informatik, Uni Hamburg

Overview

- **The Quest for Reliability**
- **Some areas with deficiencies we worked on**
  - Role of sampling
  - Feature modeling
  - Representation and irregular pyramids
- **Conclusions**

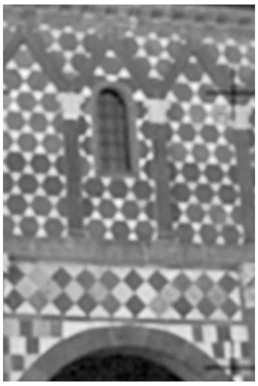
The Quest for Reliability

- **What do I mean by reliability?**
  - no formal definition yet
  - stability: small change in the data leads to small change in the result
    - no drastic changes in number of objects, neighborhood and geometry e.g. upon slight change of viewpoint or scale
  - robustness: irrelevant changes do not change the result
    - illumination, noise
  - avoid error classes that are difficult to “repair”
  - criteria to predict quality of results
    - classify images / problems according to difficulty
    - know the limits of low-level analysis, but
    - extract as much information from the original data as possible
- **Subsequent algorithms become simpler, faster and less application specific**

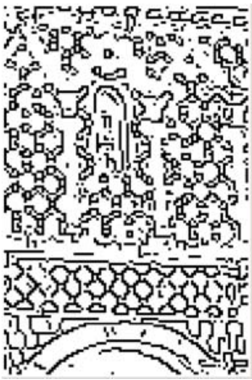
Examples for Lack of Reliability (1)

- **Edge detection so bad that it becomes useless**

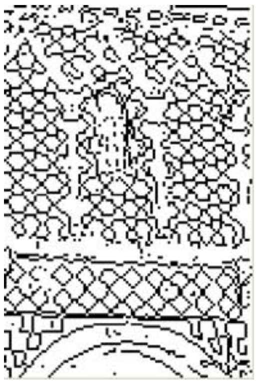
Original (detail)



Canny’s alg. (local maxima of Gaussian gradient)



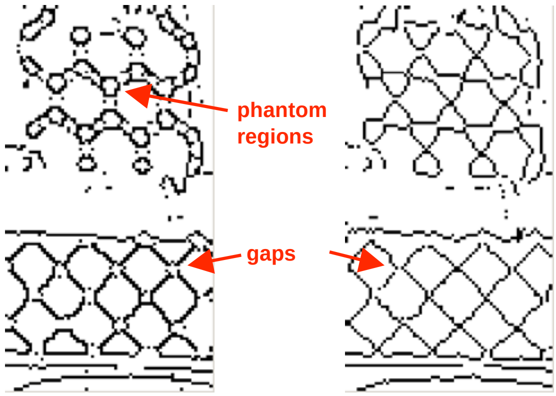
Shen-Castan’s alg. (zero crossings of diff. of expon.)



Examples for Lack of Reliability (2)

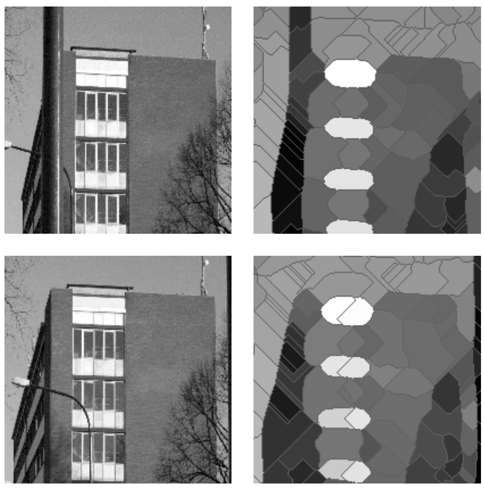
Topology of contours is incorrect

- region merging due to gaps in edges
- phantom regions at saddle points
- general: bad junction representation, contrary to their high perceptual significance



Examples for Lack of Reliability (3)

- **Slightly different viewpoint, but quite different segmentations (here: using the watershed algorithm)**
- **similar phenomenon: merging of regions essentially unpre-dictable (depends on subtle gray-value properties)**

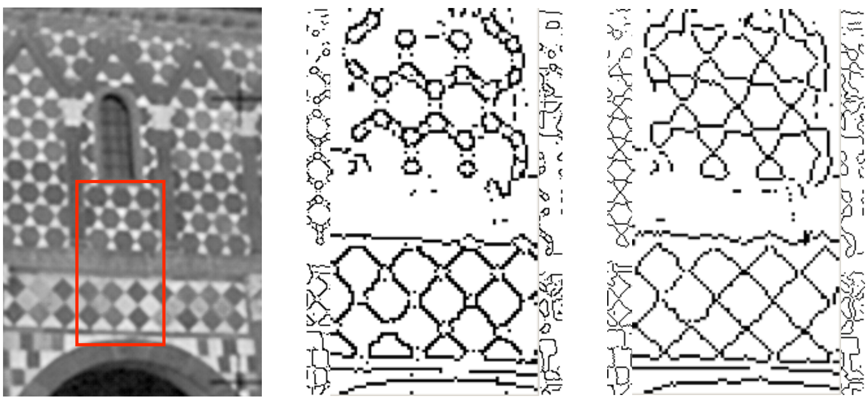


The Role of Sampling

- **Theories are often in the continuous domain**
- **But computers work in the discrete domain**
- **Sampling changes the data rather drastically**
- **But little is known about the effect of sampling in image analysis**
  – results from other fields cannot simply be transferred
  – Shannon's sampling theory:
    - measures function similarity (L2-norm)
    - uses band-limiting as criterion
  – discretization errors analysis from numeric's
    - give asymptotic error for grid refinement, not absolute error
  – need geometrical / topological sampling theorem

Empirical Result: Interpolation Helps

- **Twofold oversampling (e.g. via interpolation) improves segmentation results significantly**



Sampling Rate (1)

(Köthe: DAGM 2003)

- **Shannon's sampling theorem:**

  $f(x_1, x_2)$ can be reconstructed from samples at distance λ_N

  iff Fourier transform $F(\omega_1, \omega_2)_{f*g} = 0$ for $|\omega_1|, |\omega_2| > \omega_N = \pi / \lambda_N$
  (band-limited function)

- **if the image is band-limited, a linearly filtered version is band-limited as well (at same or smaller frequency)**

  because $f * g \circ\!\!-\!\!\bullet\ F\,G$ **(convolution theorem)**

⇨ $f_x = f * \frac{\partial}{\partial x} g$ **can still be sampled at** λ_N

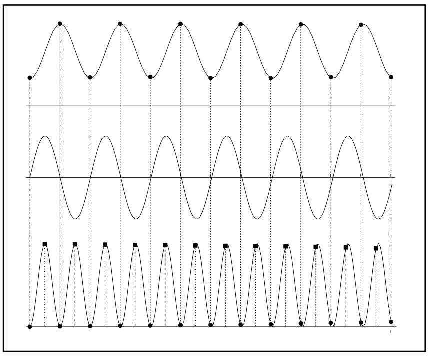
Sampling Rate (2)

- **structure tensor calculation involves products of functions, and** $f * g \circ\!\!-\!\!\bullet\ F\,G$ **(modulation theorem)**
- **support of** $F * G$ **is morphological dilation of the supports of F and G**

⇨ f_x^2 **has band-limit** $2\omega_N$

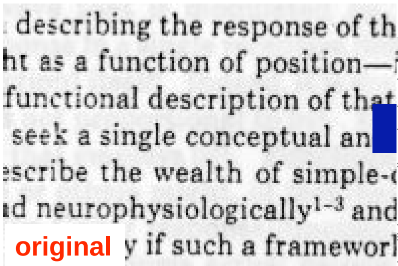
must be sampled at $\frac{\lambda_N}{2}$

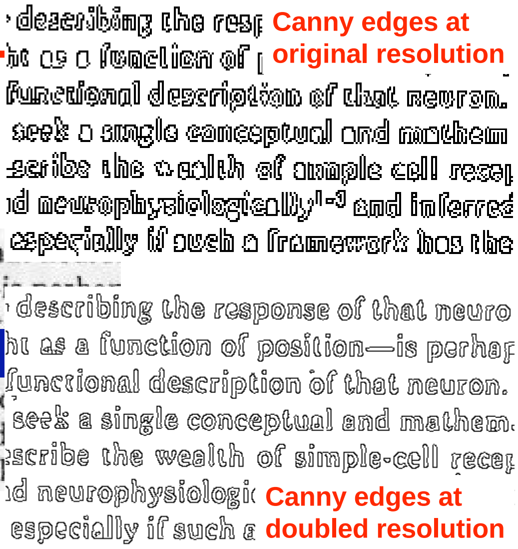
otherwise information is lost



Sampling Rate (3)

The sampling problem is very noticeable!

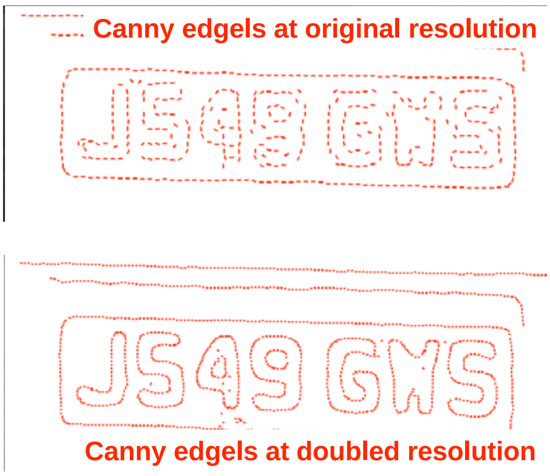




Sampling Rate (4)

Sub-pixel edgel resolution doesn't help either

original
(line width ~ 2 pixel)



Doubling of the Sampling Rate

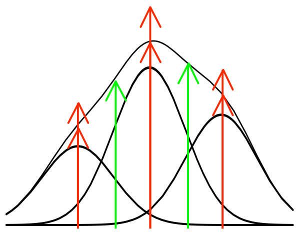
- **Not: interpolation of original image & standard filtering**
- **But: convolution of a discrete function with a continuous filter**

$$(f * g)(x,y) = \sum_{i,j} f(i,j)g(x-i, y-j)$$

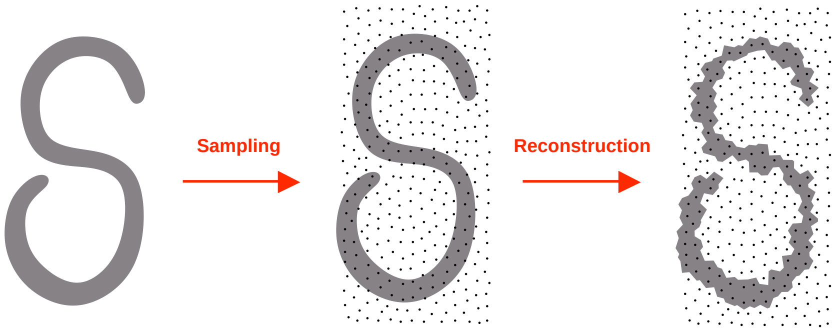
is a *continuous function*

⇨ **can be trivially sampled at twice the original sampling rate**

- **thus, double resolution during gradient calculation,**



Towards a Geometric Sampling Theory
Step 1: Binary Images



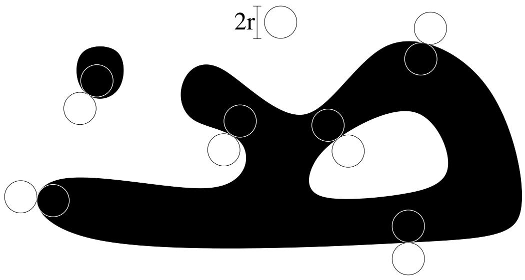
How can we ensure that shape characteristics are preserved in the digital image, independent of grid type, rotation and translation?

Prior Work

- **we directly build upon work of Serra (1982), Pavlidis (1982), Ronse and Tajine (2000), Latecki et al. (1998)**
- **geometric sampling theorems based on the notion of *r-regular shapes*:**

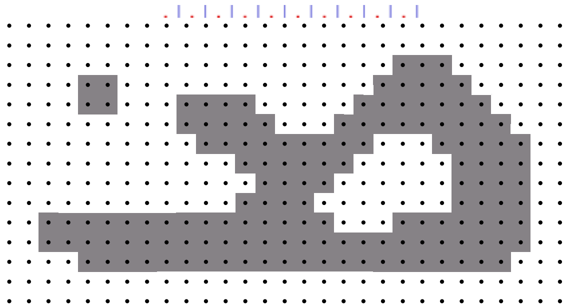
osculating r-balls at each boundary point of the shape

⇨ curvature bounded by 1/r
⇨ shape is invariant under opening and closing with r-ball structuring element (bounded shape diameter)



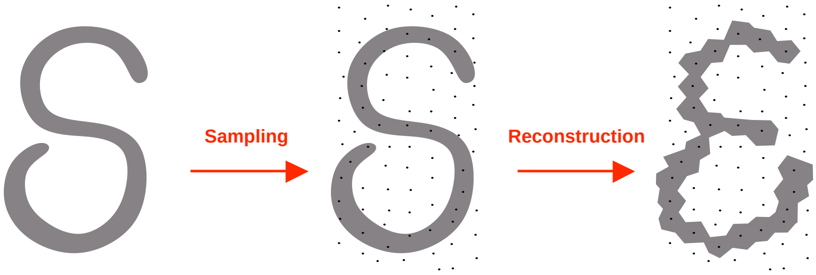
Sampling Theorem of Pavlidis (1982)

- **Reconstructed shape is topologically equivalent if**
  - original shape is r-regular
  - square grid with sample distance at most $\sqrt{2}\,\mathbf{r}$ is used
  - subset digitization is used



Problems with Shape Similarity Definition

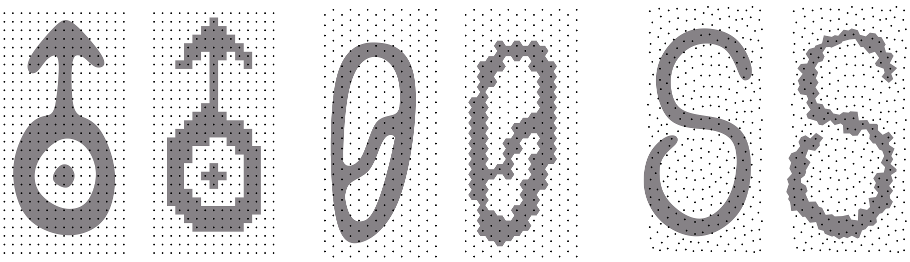
- **Shape similarity criteria sometimes fail:**
  - topological equivalence (Pavlidis, Latecki & Gross)
  - equivalence of homotopy trees (Serra)
  - small Hausdorff distance (Ronse & Tajine)



Strong r-Similarity

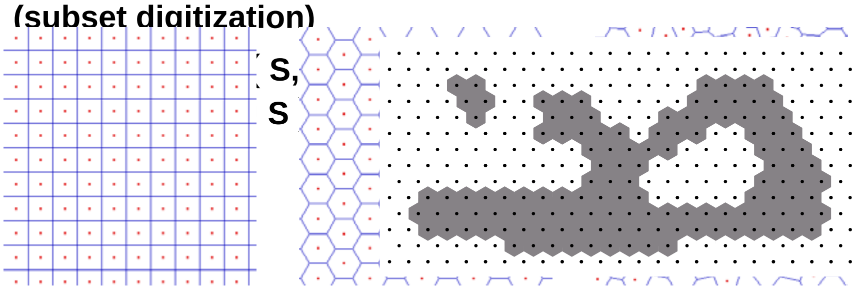
- **Why do other criteria fail?**
  - topological criteria: arbitrary geometric displacements allowed
  - Hausdorff distance: arbitrary (non-unique) point correspondences
- **restrict both the topological and geometric distortion**
- **strong *r*-similarity: bounded homomorphism**

$$\in R^2 : \quad |\mathbf{f}(\vec{p}) - \vec{p}| < r \quad \textbf{(r-homomorphism)}$$



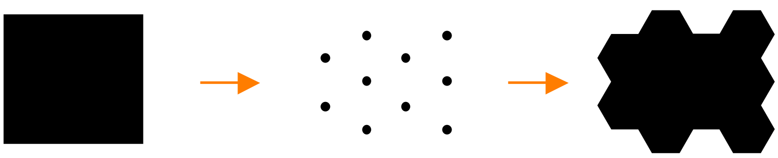
Digitization and Reconstruction

- *r*-grid: countable point set $G \subset \mathbb{R}^2$ where radius of Voronoi regions $\leq r$
- Pixel(p) : Voronoi region of point $p \in G$
- Digitization(S, G) : { $p \in G \cup S$ }
  (subset digitization)
- S,
  S



More Realistic Camera Model (1)

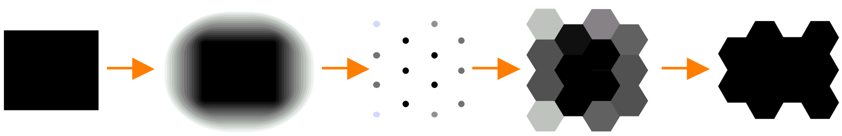
- **the camera model so far**
  - assume that the ideal (pinhole) camera image is a binary shape
  - subset digitization of the binary shape
  - reconstruction by filling the pixels



More Realistic Camera Model (2)

more realistic model: real cameras blur the analog image with their point spread function ⇨ analog image no longer binary

– assume that the ideal (pinhole) camera image is a binary shape
– blur analog binary shape with PSF
– digitize blurred image (delta functions at sampling points)
– reconstruct gray-scale image
– reconstruct binary shape by thresholding the gray-scale image (as in Latecki et al. 98)

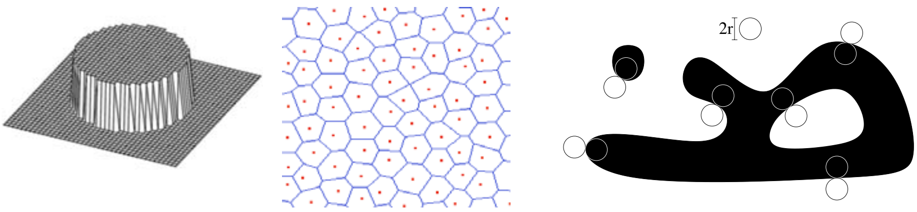


Sampling Theorem for Blurred Images

(Stelldinger & Köthe: DGCI 2003, DAGM 2003)

Reconstructed shape is (strongly) (*r* ’+*p*)-*similar* if

– points spread function is disc with radius *p*
– subset digitization with arbitrary *r*’-grid is used
– original shape is *r*-regular with **$r' + p < r$**



- **Similar to result of Latecki et al. 98, who used square grid and PSF that exactly matches the pixel shape**

Sampling of Grayscale and Color Images (1)

(Stelldinger; IWCIA 2003)

- **How to define?**
  - topological equivalence? r-similarity of all level-sets?
- **How to proof under realistic assumptions?**
  - partial results for r-regular level sets, but what about corners?
- **How to use in practice**
  - is there a topological low-pass filter (analogous to Shannon's)?
  - closest candidate: iterative average of opening and closing

Sampling of Grayscale and Color Images (2)


regularized original


nearest-neighbor rec.


linear reconstruction

Limitations of Feature Models

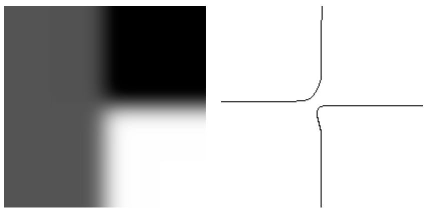
- **Topology is more fundamental than geometry!:**
  - partitioning of the plane requires to deal with regions, edges, and corners/junctions simultaneously
  - regions and contours must be duals
- **Image analysis doesn't care much about topology:**
  - regions, edges and junctions are detected separately and only integrated afterwards
  - topological errors of the separate detectors are hard to detect and repair
  - in practice, unreliable heuristics used for integration
- **too few systematic investigations of topological errors in the context of image segmentation**

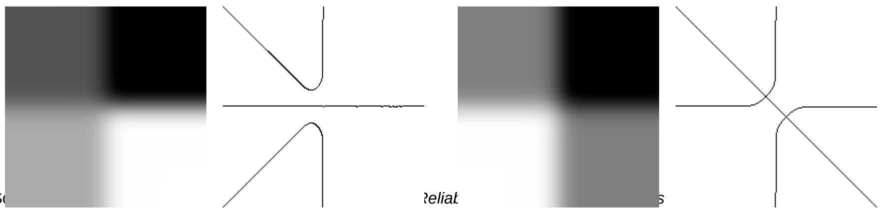
Errors at Junctions (1)

Analysis of zero-crossing algorithms

e.g. H. Neumann '88,
Deriche & Giraudon '93

- zero crossings form closed contours
- ➪ T-junctions impossible
- gaps and phantom edges
- complex dependence from contrast and scale (Florack '00)

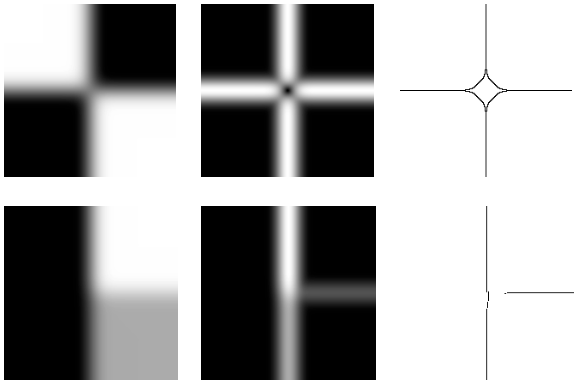




Errors at Junctions (2)

Errors of Canny's Algorithm (oriented gradient maxima)
(Rohr, Frantz, Hartkens '92-'97, Beymer '91, Rothwell et al. '94,)**:**

- – gradient minimum at saddle points
- ➯ phantom edges and regions
- – edge orientation undefined in 2ρ-neighborhood around junctions
- ➯ gaps
- – unreliable heuristic repair rules



Integrated Detection of Edges and Junctions

- **Simultaneous detection of different feature types**
- ***Contour strength* instead of *edge* and *corner strength***
- **Decomposition into feature types at the end**
- **Tensor based approaches**
  - – improved structure tensor
    - non-linear integration of the structure tensor
  - – boundary tensor
    - rotationally invariant quadrature filters using the Riesz transform

Structure Tensor Definition

- **Image gradient by Gaussian derivatives**

$$f_{x,\sigma} = f * \frac{\partial}{\partial x} g_\sigma \qquad f_{y,\sigma} = f * \frac{\partial}{\partial y} g_\sigma \qquad \text{with} \qquad g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- **Gradient tensor:**

$$Q_\sigma = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix} = \begin{pmatrix} f_{x,\sigma}^2 & f_{x,\sigma} f_{y,\sigma} \\ f_{x,\sigma} f_{y,\sigma} & f_{y,\sigma}^2 \end{pmatrix}$$

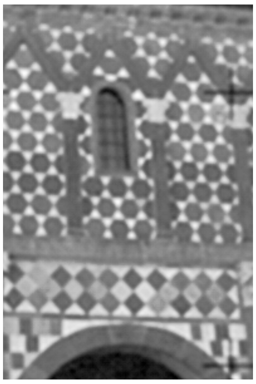
- **Structure tensor: component-wise spatial averaging of gradient tensor (calculation of a scatter matrix)**

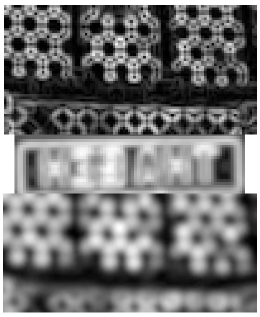
$$S_{\sigma,\sigma ©} = \begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix} = g_{\sigma'} * \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix} \qquad s_{ij} = g_{\sigma'} * q_{ij}$$

Tensor Averaging (1)

- **linear averaging causes blurring, despite the oversampling**

$$S_{\sigma,\sigma'} = g_{\sigma'} * \begin{pmatrix} f_{x,\sigma}^2 & f_{x,\sigma} f_{y,\sigma} \\ f_{x,\sigma} f_{y,\sigma} & f_{y,\sigma}^2 \end{pmatrix}$$





gradient sq. magnitude

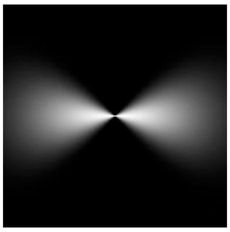
trace of structure tensor (original resolution)

trace of structure tensor (doubled resolution)

Tensor Averaging (2)

(Köthe: DAGM 2003)

- Solution: use a non-linear filter
  ➪ tensor averaging only along edges
- experimented with different possibilities (tensor differential equations, elliptic and polar separable kernels)
- best results with hour-glass kernel
- expresses likely continuations of the edge



$$h_{\sigma',\rho}(r,\varphi,\varphi_0) = \frac{1}{N} e^{\frac{-r^2}{2\sigma'^2}} e^{\frac{-\tan^2(\varphi-\varphi_0)}{2\rho^2}}$$

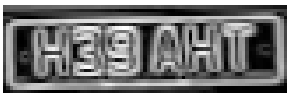
rotate filter according to local edge orientation (minor tensor axis)

Tensor Averaging (3)

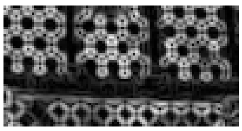
- Filter definition:

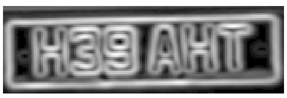
$$t_{ij}(x,y) = \sum_{x',y'} h_{\sigma',\rho}\left(x-x', y-y', \vec{n}(x'y')\right)\, q_{ij}(x',y') \quad i,j \in \{1,2\}$$

- Results – no blurring, improved junction response:

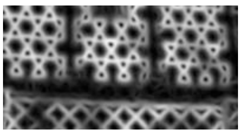


gradient sq. magnitude



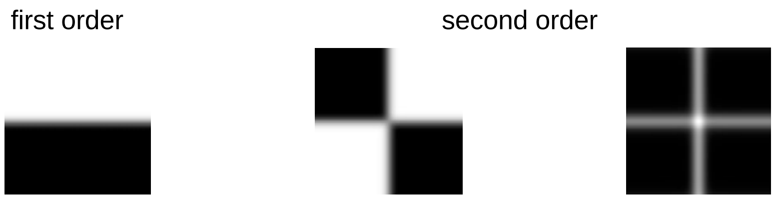


tensor trace after non-linear averaging



Combination of 1st and 2nd Order Filters (1)

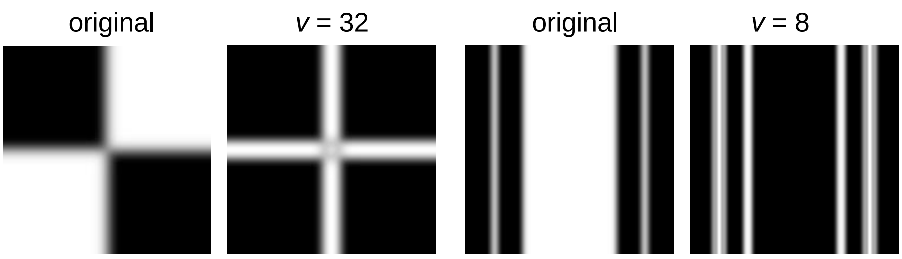
- **use different filters to be sensitive for different feature types:**



- **integrate the filter responses into a common tensor representation ⇒** boundary tensor
- **but: make sure that the filters combine reasonably everywhere ⇒ quadrature filters, not derivatives**

Combination of 1st and 2nd Order Filters (2)

- **straightforward combination:** $\mathbf{T} = \nabla I \nabla I^T + v H^2$
- **but: How to chose *v*?**
  - in general, no single *v* works well on the entire image
  - often responses are not unimodal
  - ➪ try to eliminate *v*

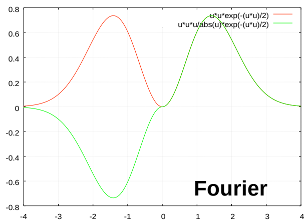


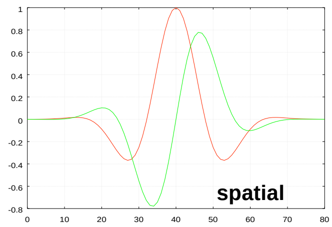
Combination of
1st and 2nd Order Filters (3)

- **Established method in 1D: quadrature filters**
  - pair of even and odd filters with same amplitude spectrum (derivatives have different spectra!)

Hilbert Transform
(in Fourier domain)

$$\mathbf{K_{odd}} = \frac{j\omega}{|\omega|}\mathbf{K_{even}} = j\,\mathbf{sign}(\omega)\mathbf{K_{even}}$$

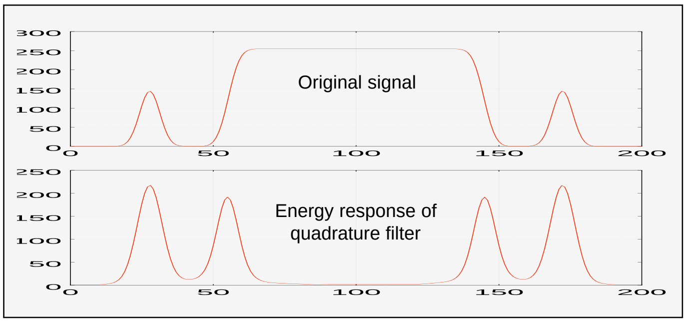




Combination of
1st and 2nd Order Filters (4)

- **combined energy of even and odd filters is the same for 1st and 2nd order structure (step edges and ridges)**

$$\mathbf{E_{quad}} = (\mathbf{k_{odd}} * \mathbf{f})^2 + (\mathbf{k_{even}} * \mathbf{f})^2$$



Combination of 1st and 2nd Order Filters (5)

- **How to generalize quadrature filters to 2D?**
  - usually: define a family of oriented 1D filters
  - but: How many orientations? How to combine filter responses, or select the most salient one?
  - ⇨ new tuning parameters necessary
  - ⇨ prefer parameter free, rotationally invariant generalization
- **Generalize Hilbert transform to Riesz transform**

Riesz transform (in Fourier domain)

$$K_{Riesz} = \frac{j\vec{\omega}}{|\vec{\omega}|} K_{Original}$$

compare with gradient:

$$K_{Gradient} = j\vec{\omega}\, K_{Original}$$

2D Quadrature Filters

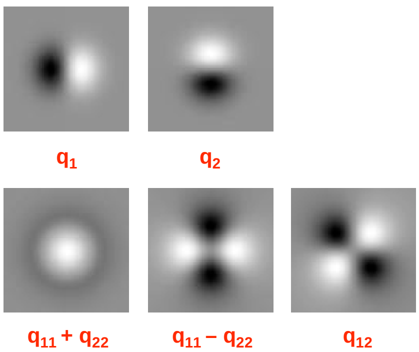
(Köthe: ICCV 2004, upcoming tensor book)

- **Define 1st order Riesz vector and 2nd order Riesz matrix from Laplacean of Gaussian filter**
- **behave like 1D quadrature filters for 1D features (edges, lines)**
- **respond to 2D features**

$$Q(\vec{\omega}) = |\vec{\omega}|^2 e^{-|\vec{\omega}|^2 \sigma^2 / 2}$$

$$Q_i = j\vec{\omega}_i |\vec{\omega}| e^{-|\vec{\omega}|^2 \sigma^2 / 2}$$

$$Q_{ik} = -\vec{\omega}_i \vec{\omega}_k e^{-|\vec{\omega}|^2 \sigma^2 / 2}$$



Boundary Tensor (1)

- **Combine filter responses into a tensor**

$$g_i = q_i * f \qquad H_{ik} = q_{ik} * f \qquad B = g g^T + H H^T$$

- **Tensor trace: boundary energy**
  - is rotationally invariant
  - signals step and line edges

Boundary Tensor (2)

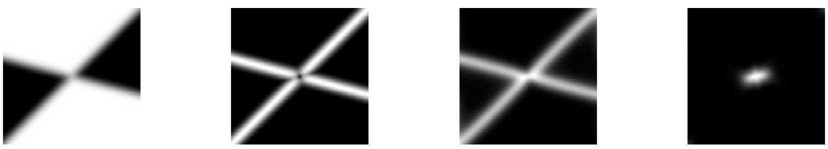
- **But boundary energy is also high at junctions**
- **Decomposition into edge and junction contributions**

$$B = B_{edge} + B_{junction}$$
$$= (\mu_1 - \mu_2)\, \vec{e}_1 \vec{e}_1^T \; + \; \mu_2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

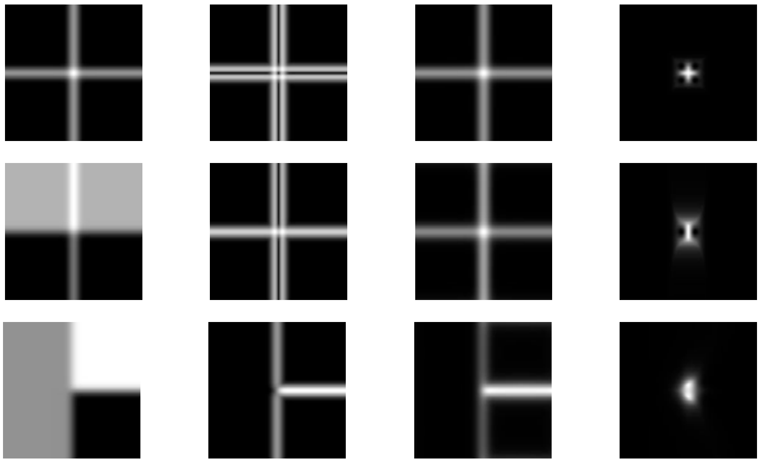
μ_1, μ_2 : large and small eigenvalues

$\vec{e}_1$: eigenvector for μ_1

- **Compute orientation of 1D structures:** $\psi = \frac{1}{2} \arctan \frac{2B_{12}}{B_{11} - B_{22}}$

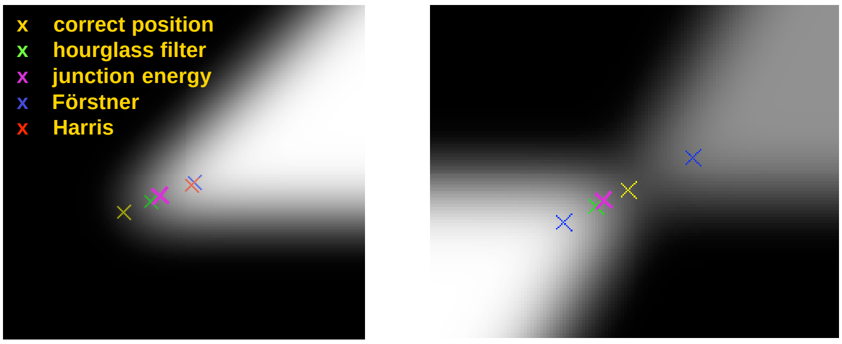


Boundary Tensor on Test Images

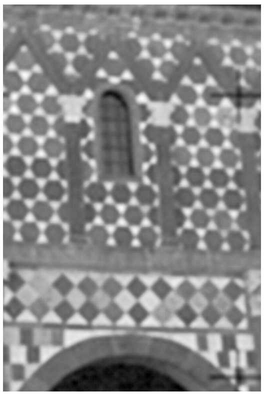


Accuracy of Corner / Junction Localization

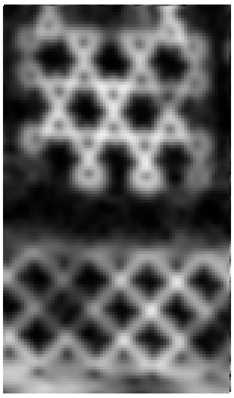
- significantly higher accuracy than traditional methods, e.g. Förstner or Harris (reduced to 30-50%)
- multiple responses much more infrequent



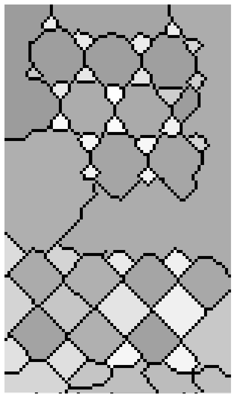
Watershed Algorithm – Boundary Tensor



Original

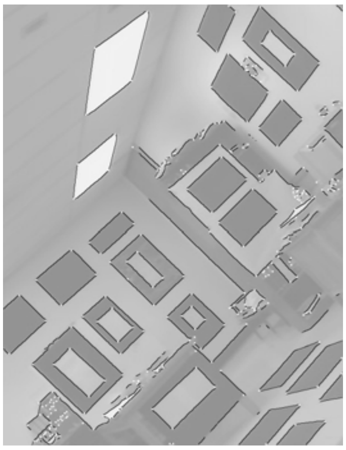


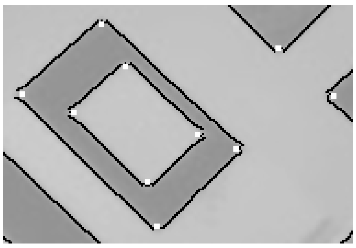
boundary energy



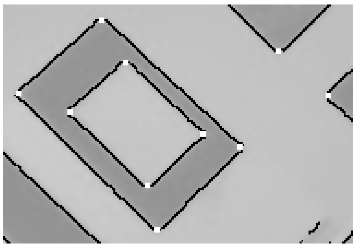
watersheds of energy

Integrated Edge/Junction Detection – Structure Tensor





edges / corners with linear averaging



edges / corners non- linear averaging

Representation of Image Partitionings

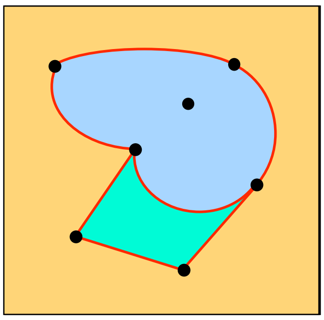
(Köthe, Meine, Stiehl: Dagstuhl 2003, DGCI 2003, BVM 2004)

- **Many algorithms create / work with image partitionings**
- **Usually each algorithm defines its own data representation**
- **disadvantages:**
  - a lot of repeated work (data structures and their consistent manipulation)
  - algorithms are difficult to combine (esp. edge and region segm.)
  - good ideas cannot be realized because data structure is too weak
  - often: heuristic solutions because proper solutions are more expensive ⇨ consistency (topology!) no longer guaranteed

⇨ **problems can be solved by a unified data representation conforming to topological requirements**

Geometric Definition of a Finite Topology

- **(geometric) partitions of the plane are the foundation of the theory:**
  - set of vertices V = {V_i: [x_i, y_i] }
  - set of open arcs A that connect the vertices A = {A_k: (0,1) → $\mathbb{R}^2$ }
  - regions R: connected components of the rest
  - V_i, A_k, R_m are the elements of the finite topological space
  - a set is open if the corresponding set in the plane is open under the Euclidean topology
  - 'open stars' basis of the topology
  - inconvenient because geometry and topology intimately tied



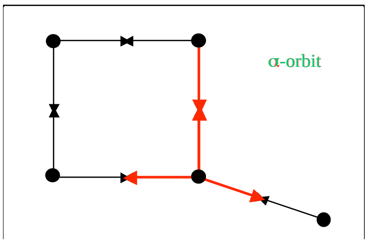
Combinatorial Map Definition of a Finite Topology

- **Combinatorial Map (Tutte '84, Dufourd/Puitg '00):**
  - set of *darts* (half-edges) D = { d_i }
  - involution α and permutation ρ of the darts (then $\rho^{-1}\alpha$ is also a permutation)
  - cycles ("orbits") of the permutations define the nodes, edges, and faces of the combinatorial map
- **Example:**

  α = {E_1=(1,1'), E_2=(2,2'), E_3=(3,3'), E_4=(4,4'), E_5=(5,5')}

  ρ = {N_1=(1,2), N_2=(1',4), N_3=(2',3), N_4=(5,4',3'), N_5=(5')}

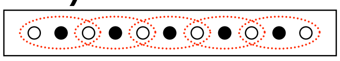
  $\rho^{-1}\alpha$ ={F_1=(1,4,5,5',3',2'), F_2=(1',2,3',4')}



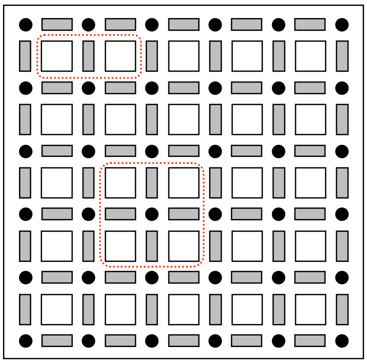
- **Advantage: explicit coding of structure of imbedding**
- **Disadvantage: faces may not have holes**

Khalimsky's Definition of a Finite Topology

- **Khalimsky's line (Khalimsky et al. '90)**
  - alternate open and a closed points

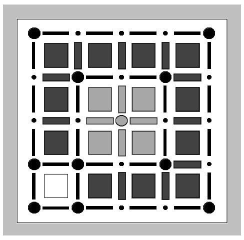


- **Khalimsky's plane:**
  - product of two such lines
  - consists of open, closed and mixed points
  - open sets as products of the open sets of the lines
- **Advantage: simple regular structure**
- **Disadvantage: can only represent regular grids**



Block Complex Definition of a Finite Topology

- **Define a new cell complex on the basis of an existing one (Kovalevsky '00):**
  - completely subdivide the cell complex into 0-, 1-, 2-blocks
  - n-block is homeomorphic to a n-sphere
  - bounding relation: if a cell from B_i bounds a cell from B_k, then B_i bounds B_k
- **Example:**
  - gray or white squares/rectangles: 2-blocks
  - small circles and black lines: 1-blocks
  - large circles: 0-blocks



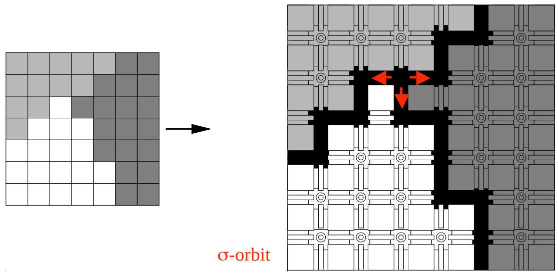
- **Advantage: can be defined on top of any finite topological space**
- **Disadvantage: n-sphere requirement is too restrictive**

Algorithms to Create a Combinatorial Map

- **Crack insertion algorithm:**
  - block complex on top of Khalimski grid
  - define darts and contains relation to go to XPMap
- **Watershed algorithm:**
  - irreducible 8-connected boundary on pixels
  - classification of boundary pixels into cell types
- **Sub-pixel watershed algorithm**
  - smooth interpolation of the image
  - find critical points and use Runge-Kutta algorithm for continuous contour following
- **all 3 representations fulfill axioms and can implement the same abstract data type**

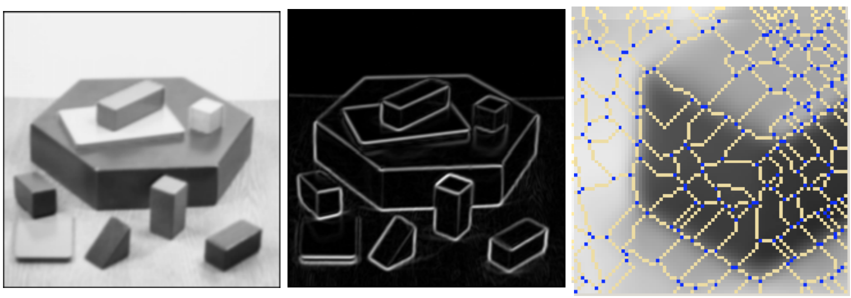
XPMap from
Crack Insertion Algorithm

- **start with 4-connected region image**
  - ➪ insert space for cracks ➪ label cracks
  - ➪ transform block labeling of Khalimksy plane
  - ➪ merge edges ➪ define orbits



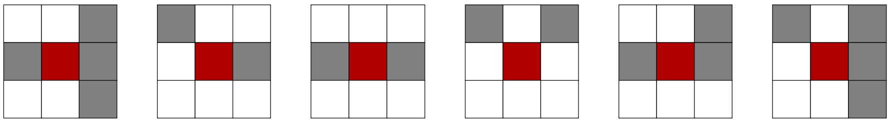
XPMap from
Watershed Algorithm (1)

- **start with boundary indicator (e.g. gradient magnitude)**
  - ➪ do non-maxima suppression with watershed algorithm, leave 8-connected irreducible boundary
  - ➪ label boundary points as vertex or edge
  - ➪ define orbits

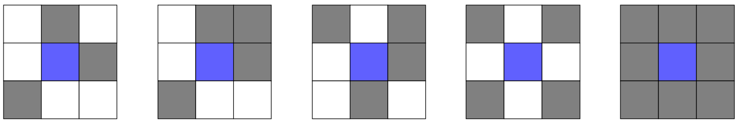


XPMap from Watershed Algorithm (2)

- **Point labeling:**
  - classify point as edge if it has exactly two continuations, e.g.

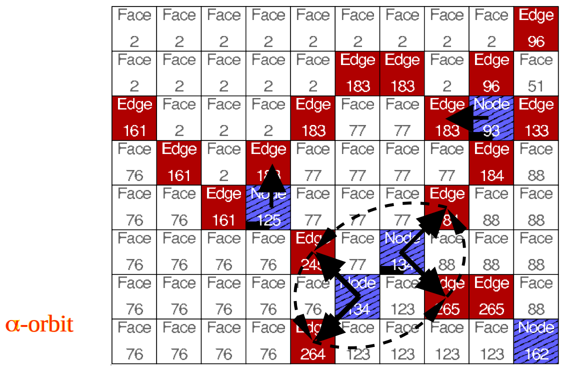


  - as vertex otherwise, e.g.



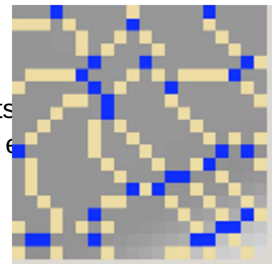
XPMap from Watershed Algorithm (3)

- **Define α-orbit by edge following**
- **Define σ-orbit by contour following around vertex**
- $\varphi = \sigma^{-1}\alpha$



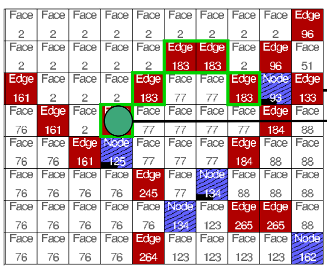
Properties of Algorithms for XPMap Definition

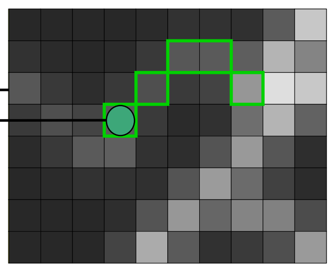
- **all 3 algorithms create topologically correct results**
- **Pixel based algorithms:**
  - \+ quite fast, use established algorithms
  - \+ easy access to pixels of all cells (vertices, edge, and regions)
  - – low geometric accuracy,
  - – big vertices (several pixels) in watershed
  - – 4-fold resolution for crack representation
- **Sub-pixel algorithm:**
  - \+ high geometric accuracy, vertices are points
  - \+ easy access to coordinates of vertices and e
  - – much slower
  - – access to region interior more difficult



The GeoMap Abstract Data Type (3)

- **Inspection of geometry and photometry: CellScanIterators**
  - – move over the pixels/points in a given cell
  - – return current coordinate ➪ can directly derive geometric properties
  - – access original or derived image at same place ➪ collect any statistic

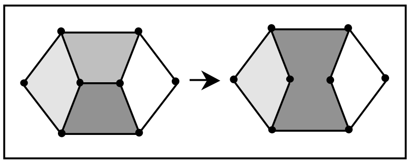




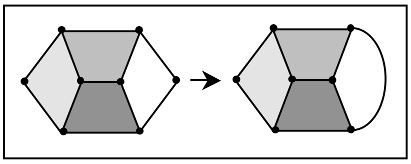
Euler Operators on the GeoMap (1)

- **Euler operators: transform an XPMap into another one**
  - Euler's relation remains valid:
  - elementary modifications: number of cells is changed by at most 1
  - complete: any transformation is concatenation of Euler operations
  - all necessary changes transparently applied to data structure, including update of user defined data (e.g. statistics)

merge faces

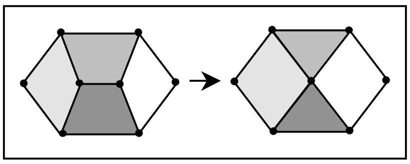


merge edges

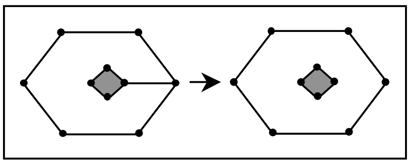


Euler Operators on the GeoMap (2)

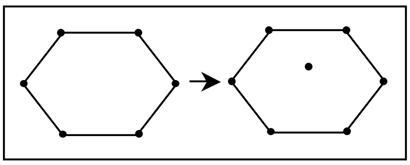
contract edge



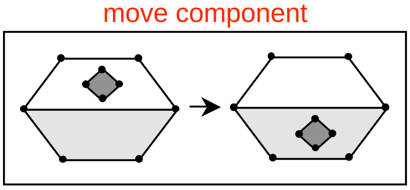
remove bridge



insert node



move component



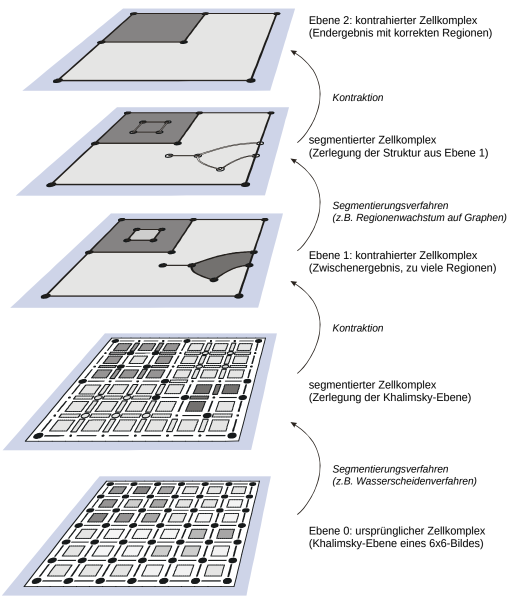
- **can prove: all operations transform a XPMap into another valid XPMap**

Euler Operators on the GeoMap (3)

- on continuous XPMap: remove cells
- on pixel-based XPMap: relabel cells
- only reductions so far

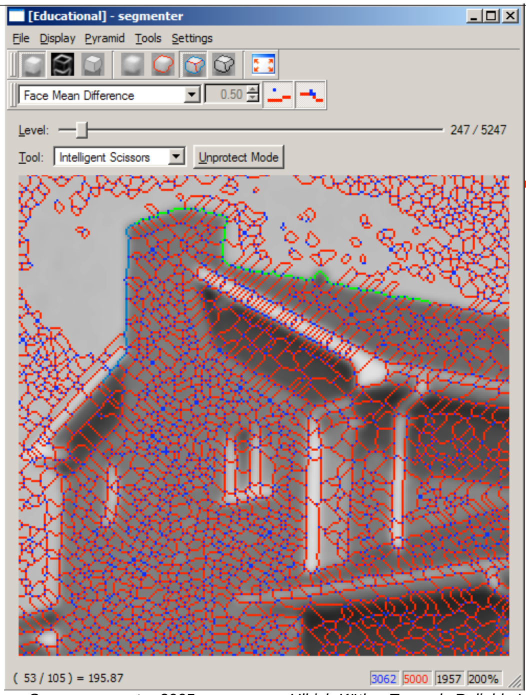
Merge regions
Merge regions
Remove bridge
Remove isolated node
Merge regions
Merge edges

Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Edge 96
Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Edge 96	Face 51
Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Node 93	Edge 133
Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Edge 184	Face 88
Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Edge 184	Face 88	Face 88
Face 76	Face 76	Face 76	Face 76	Face 76	Face 76	Edge 184	Face 88	Face 88	Face 88
Face 76	Face 76	Face 76	Face 76	Face 76	Edge 184	Face 88	Face 88	Face 88	Face 88
Face 76	Face 76	Face 76	Face 76	Edge 184	Face 88	Face 88	Face 88	Face 88	Node 162



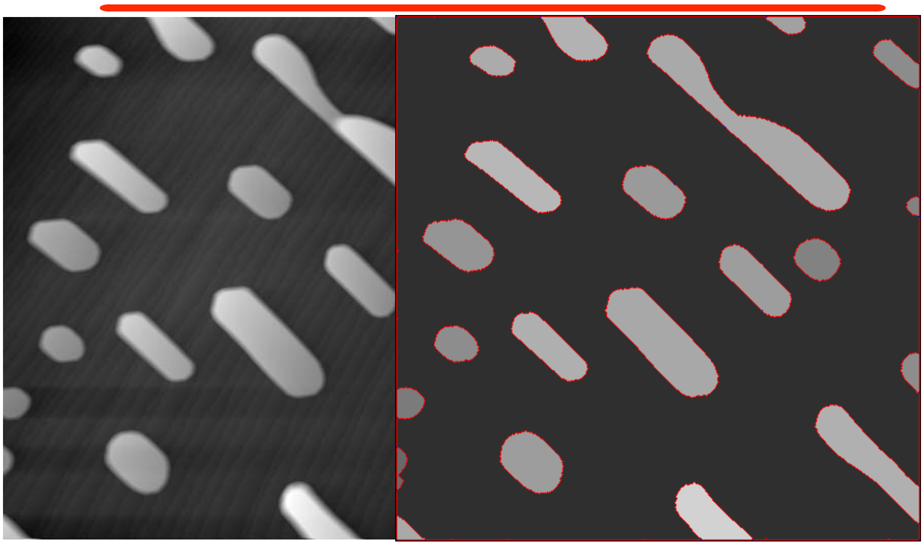
regular Pyramids sing the GeoMap

- Given: an XPMap
- Use application-specific criteria to define k-segments
- Reduce the segments into a simpler XPMap
- Repeat until desired result is achieved
- Different criteria can be used at every level
- Can build both regular and irregular pyramids (depends on criteria to define segments)

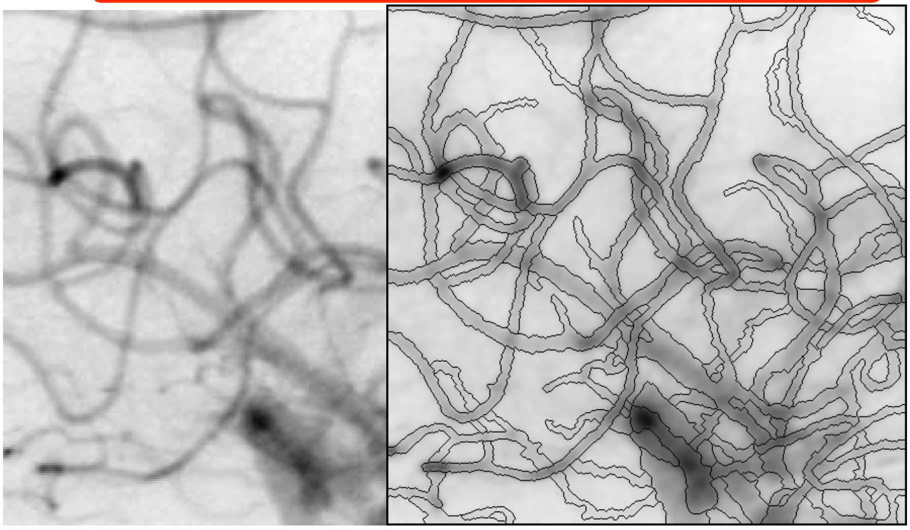


Demo

Results (1): Automatic Edge Remover



Results (2): Interactive Paintbrush



Conclusions (1)

- **Noise and low contrast cannot always be blamed for bed results in low-level segmentation**
- **We do not fully understand many crucial parts of the process**
  - How much information is preserved after discretization?
  - What's an appropriate feature model?
  - How to avoid topological inconsistencies?
  - How to evaluate algorithms performance?
  - and many more…
- **We are quick with invoking non-generic *knowledge* to avoid or repair errors**
- **But much of the information in the *data* is ignored**

Conclusions (2)

What I've shown

- **Oversampling prevents information loss**
- **Under certain conditions, topology does not change during discretization**
- **Better feature models can be built using tensors**
- **Topological representation helps to**
  - eliminate heuristics
  - preserve topological correctness during all processing steps
  - combine and unify algorithms
- **GeoMap abstract data type makes life much easier**
  - programming at high abstraction level, but still fast

Conclusions (3)

- **But we have barely scratched the surface**
  - more realistic conditions for the sampling theorem and handling of more complicated shapes
  - tensors should cover all feature configurations and become less noise sensitive
  - better criteria to build irregular pyramids (e.g. gestalt, learning)
  - extension to 3D
  - systematic evaluation (experiments in batch-mode)
  - integration with higher-level vision

- **Software (VIGRA) and papers at**

  http://kogs-www.informatik.uni-hamburg.de/~koethe/papers/

Thank you!