Part C
Applications to image analysis problems

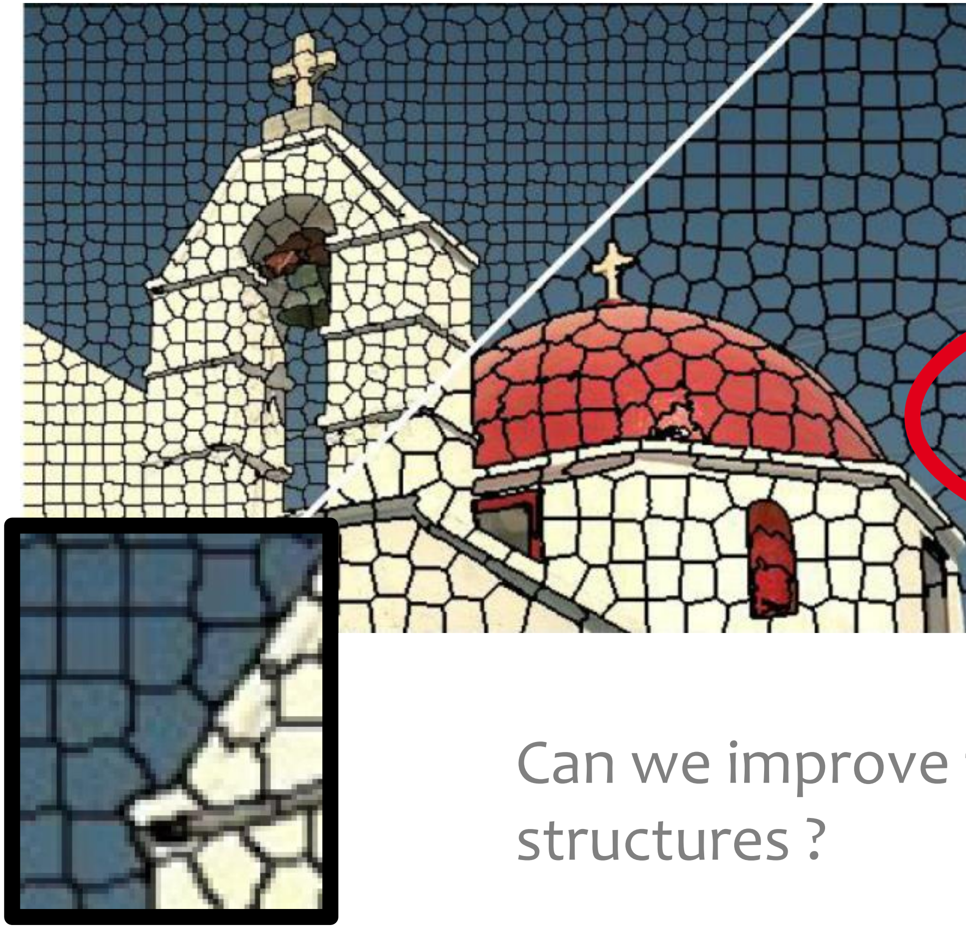
Outlines

Polygonal superpixels
- with Voronoi diagrams
- with Kinetic data-structures
- application to object contouring

Delaunay point processes
- principle
- application to object contouring
- application to line-network extraction
- application to image compression

1
Polygonal superpixels

Superpixels

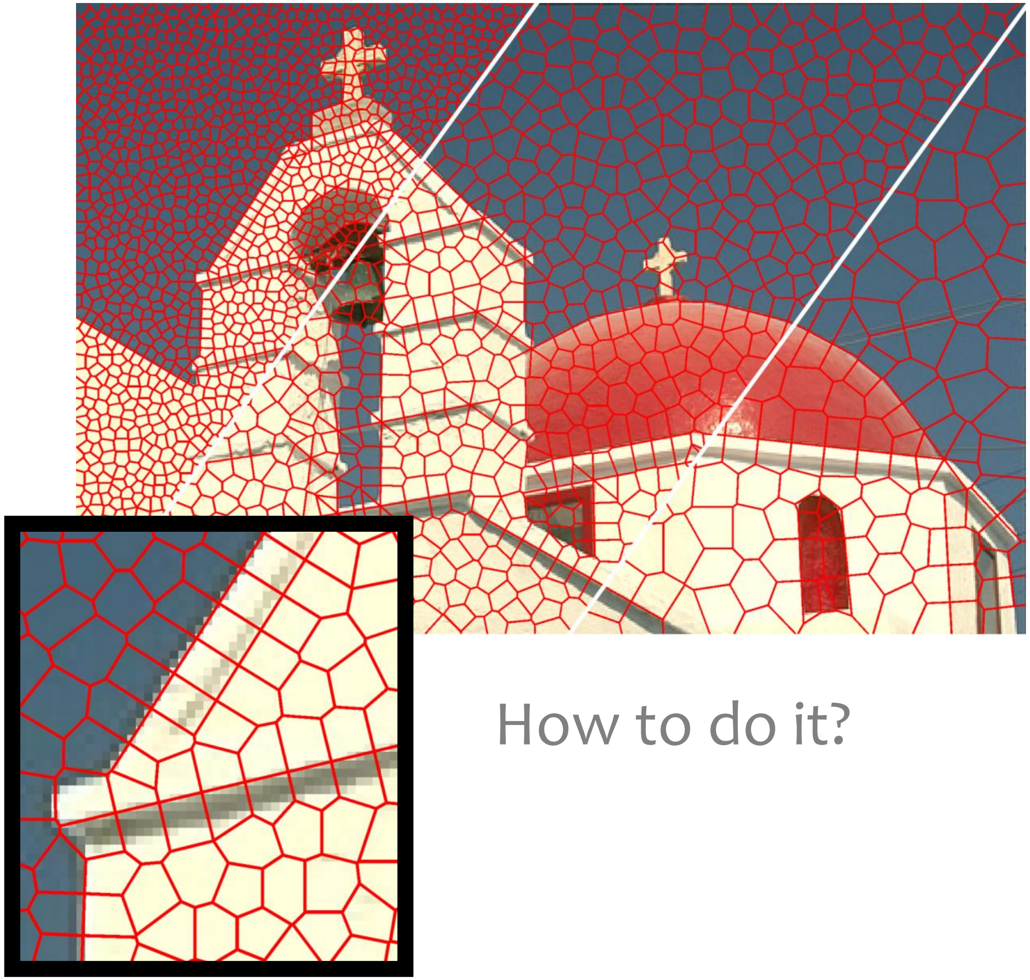


Over-segmentation of images

- ✓ contour preservation
- ✓ algorithmic complexity
- ✗ storage
- ✗ control on region shapes
- ✗ region adjacency

Can we improve this with geometric data-structures ?

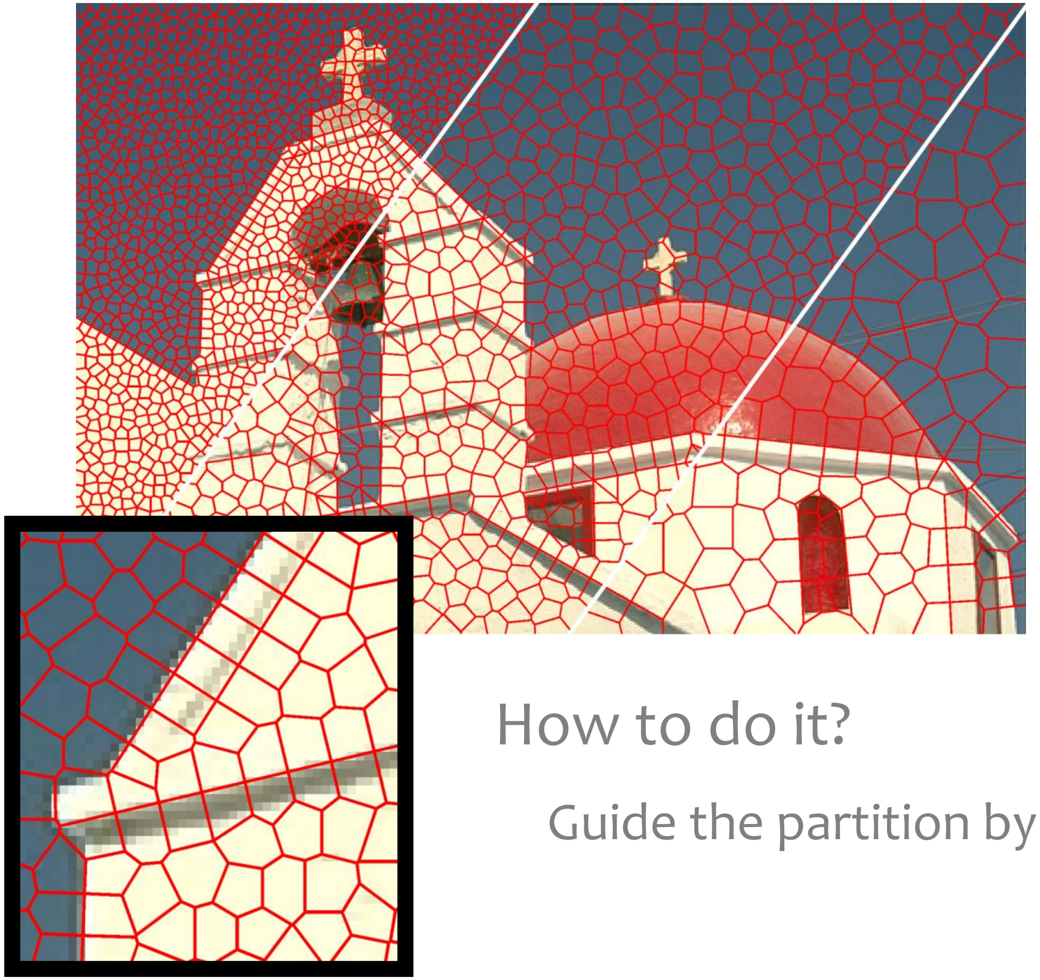
Superpixels as Voronoi cells



- ✓ storage
  (2D Delaunay triangulation)
- ✓ control on region shapes
  (convex polygons)
- ✓ region adjacency
  (uniqueness)

How to do it?

Superpixels as Voronoi cells



- ✓ storage (2D Delaunay triangulation)
- ✓ control on region shapes (convex polygons)
- ✓ region adjacency (uniqueness)

How to do it?

Guide the partition by geometric shapes

Voronoi-based Image partitioning

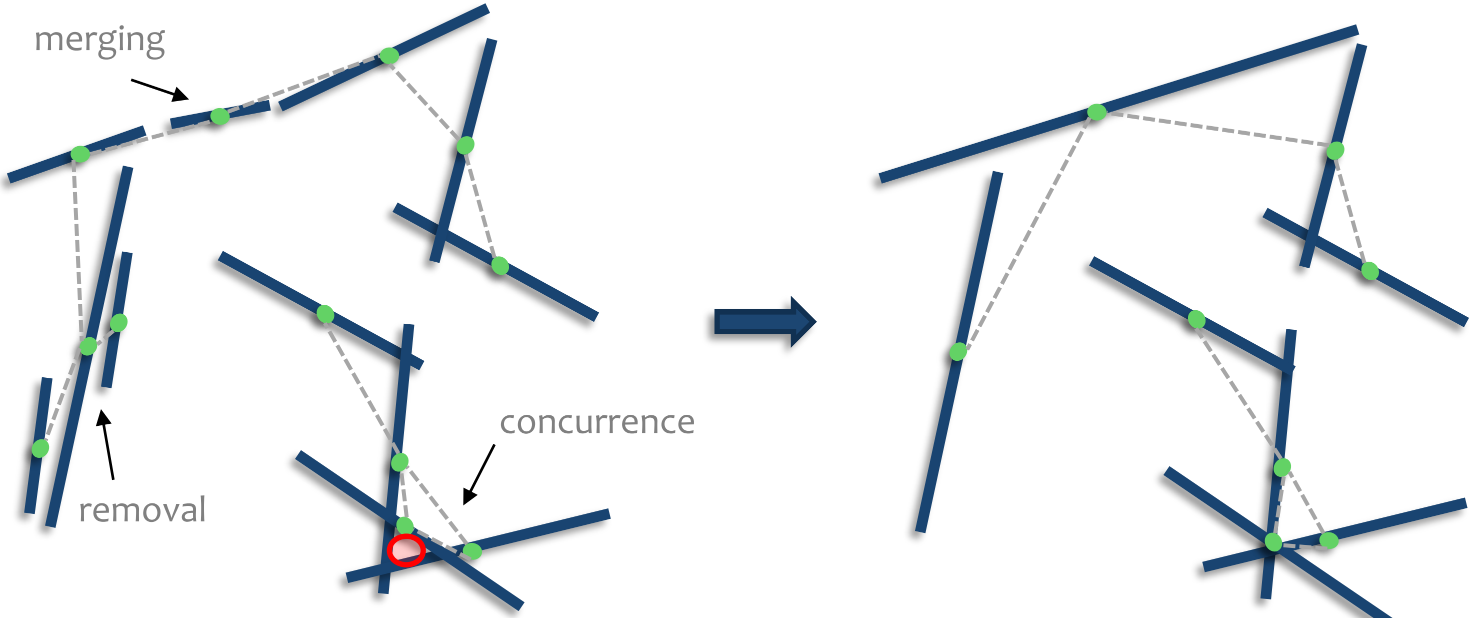
[Duan and Lafarge, Partitioning images into convex polygons, CVPR 2015]

Step 1: extraction of geometric shapes

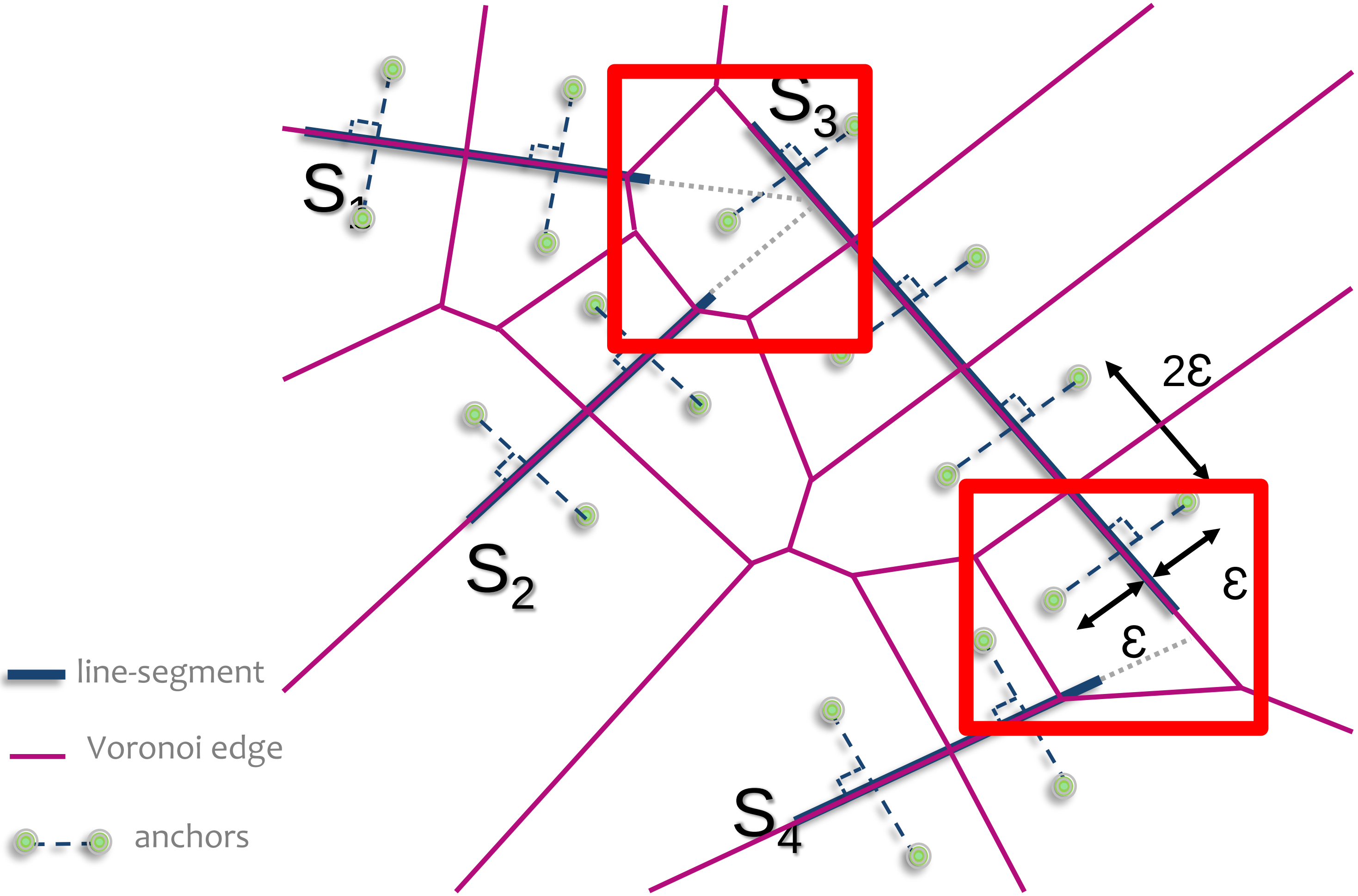
Detection of line-segments

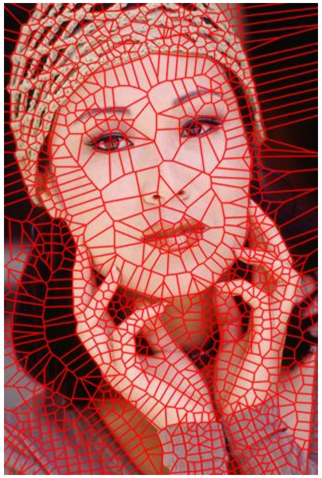
[Von Gioi et al., Lsd: A fast line segment detector with a false detection control, PAMI 2010]

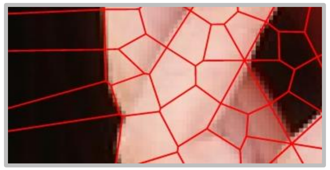
Consolidation of line-segments



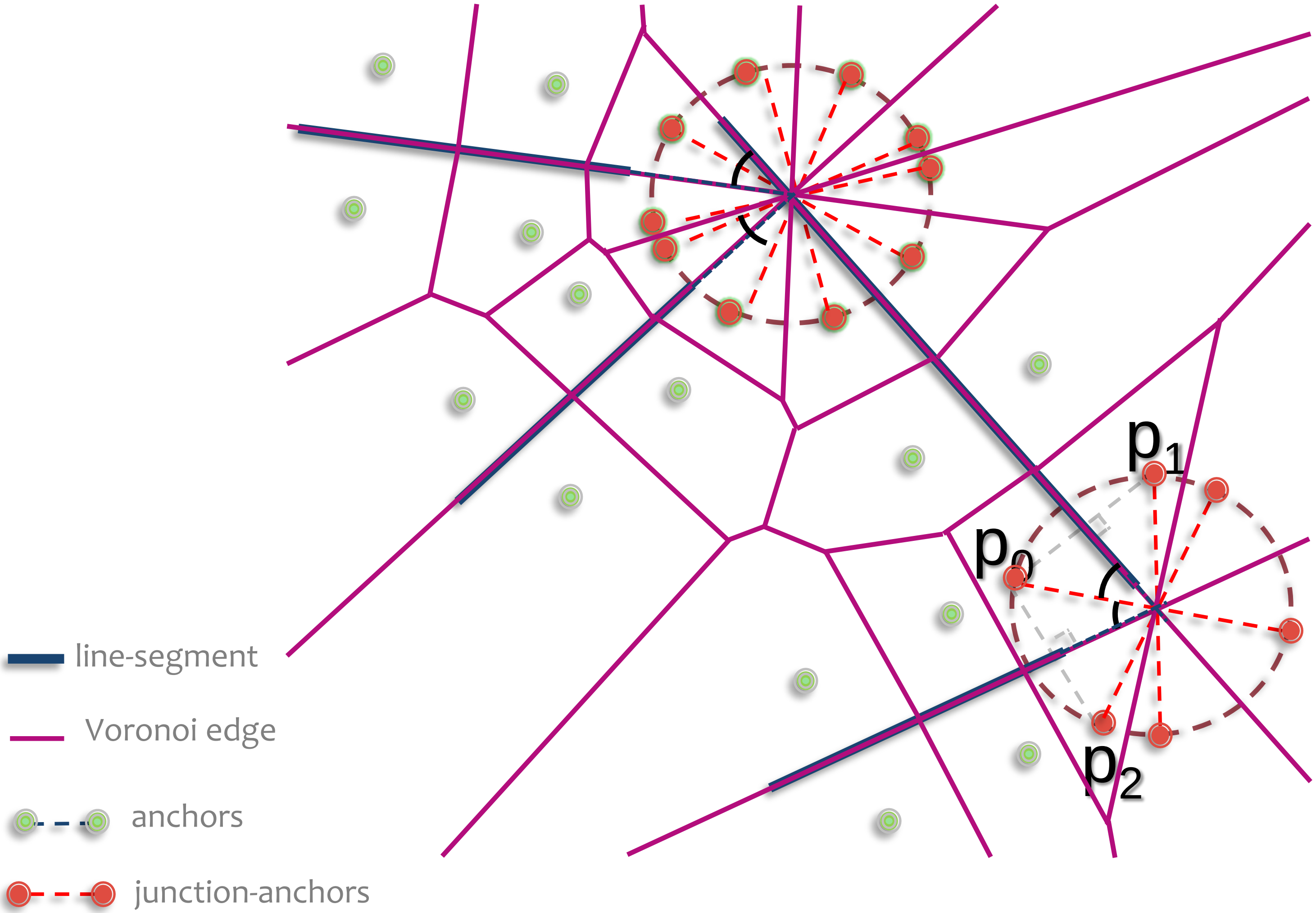
Step 2: anchoring

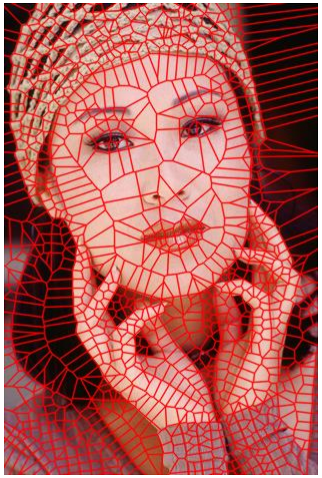


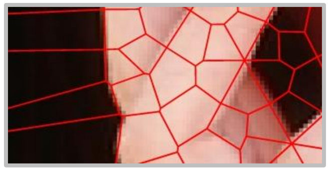




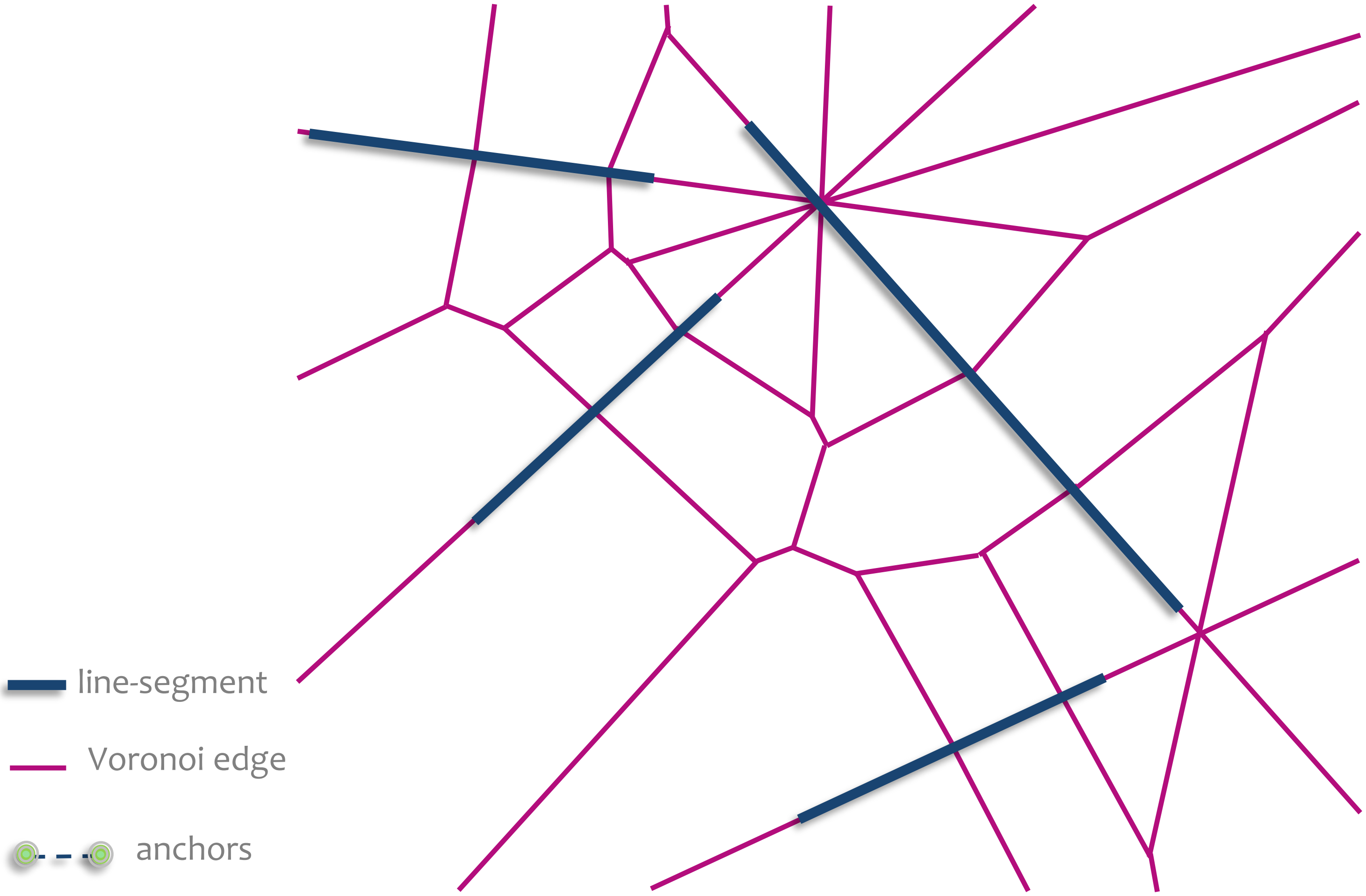
Step 2: anchoring

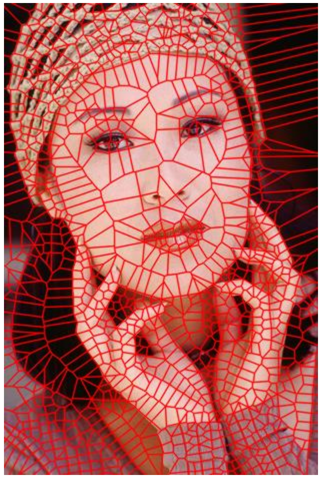


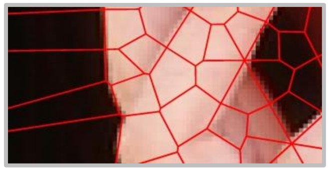




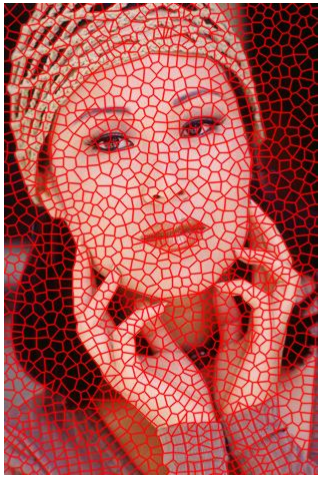
Step 2: anchoring

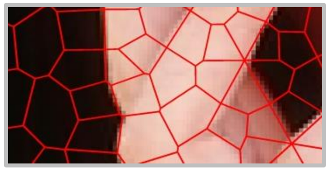


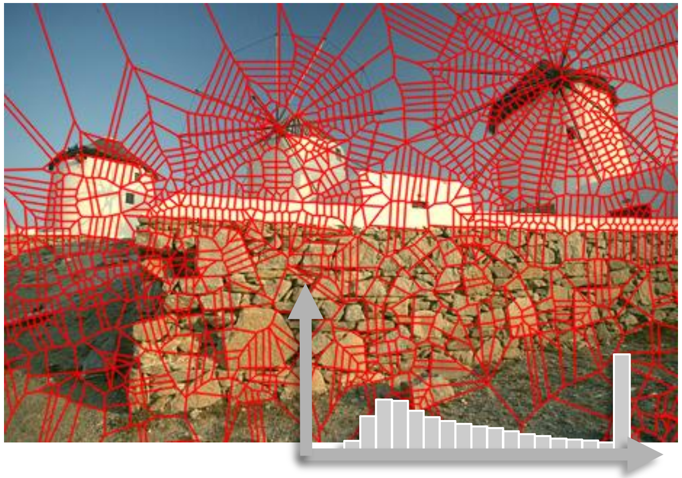




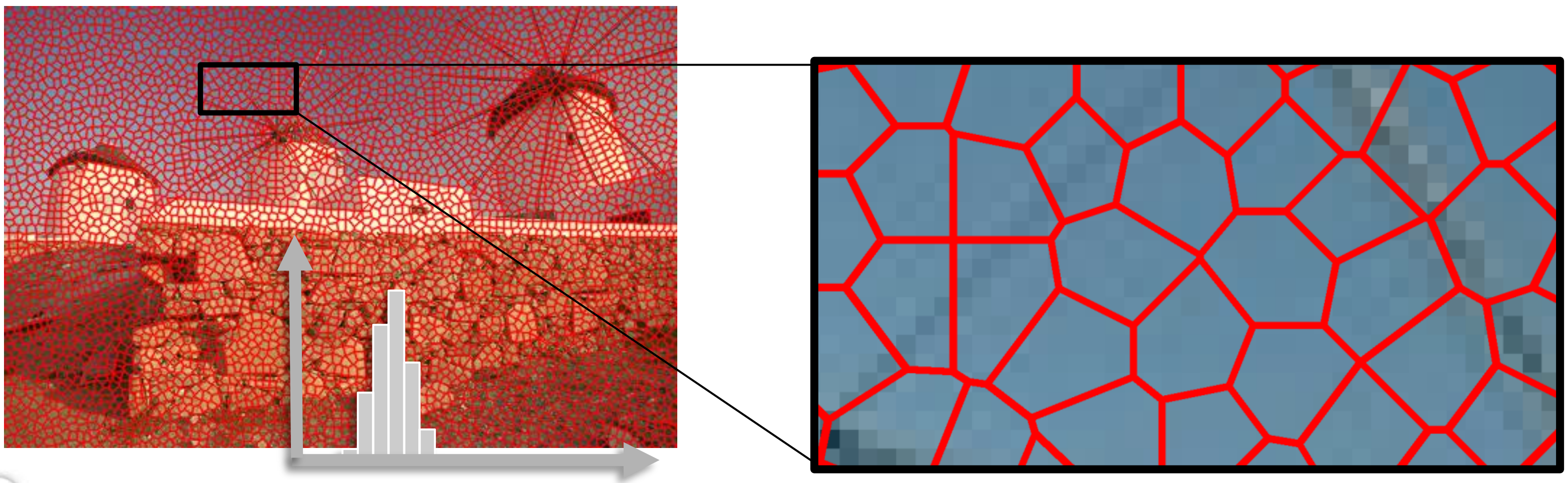
Step 3: homogeneization



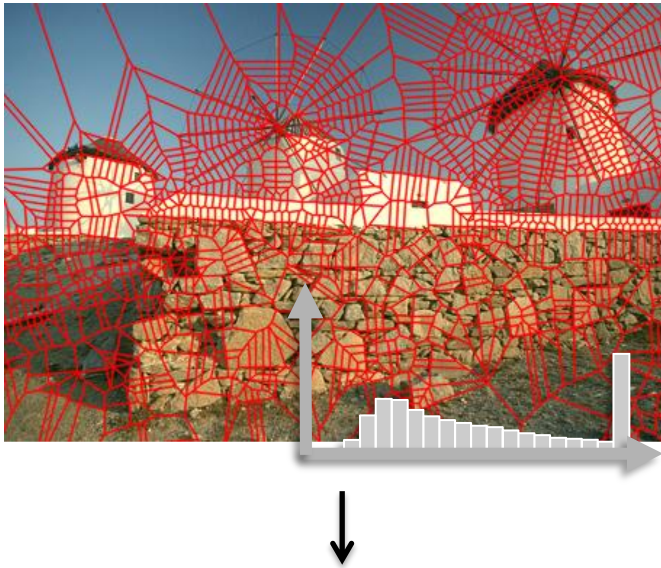


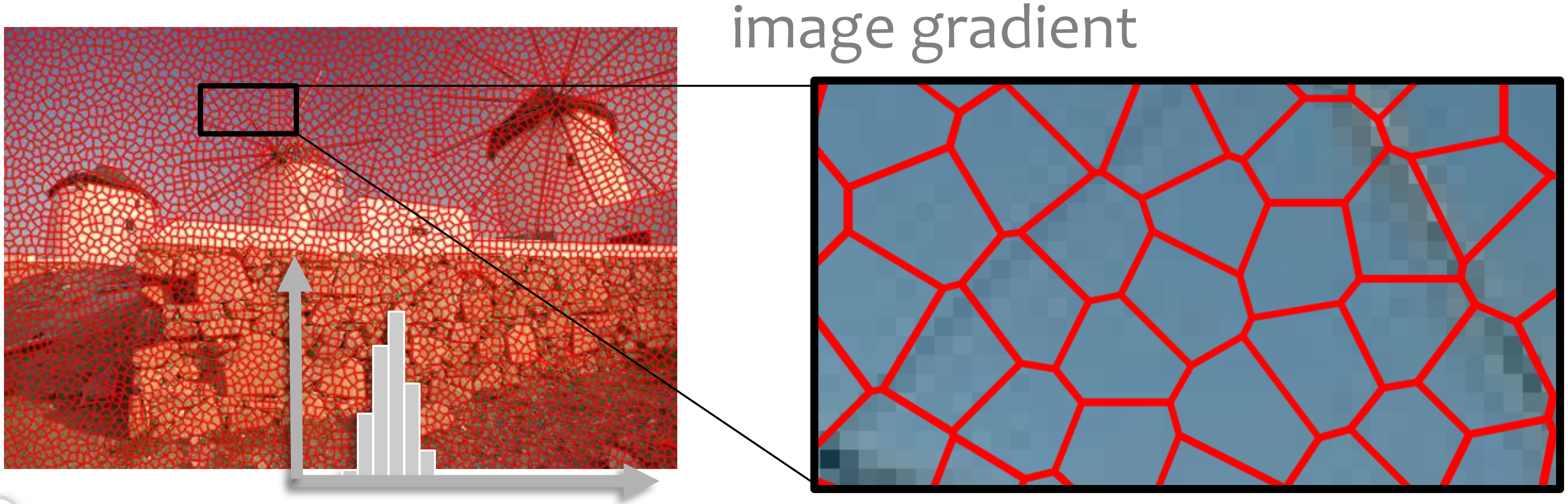


Poisson disk sampling



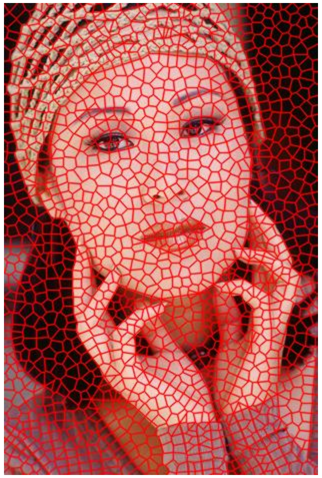
Step 3: homogeneization

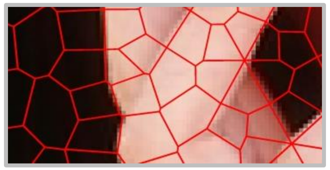


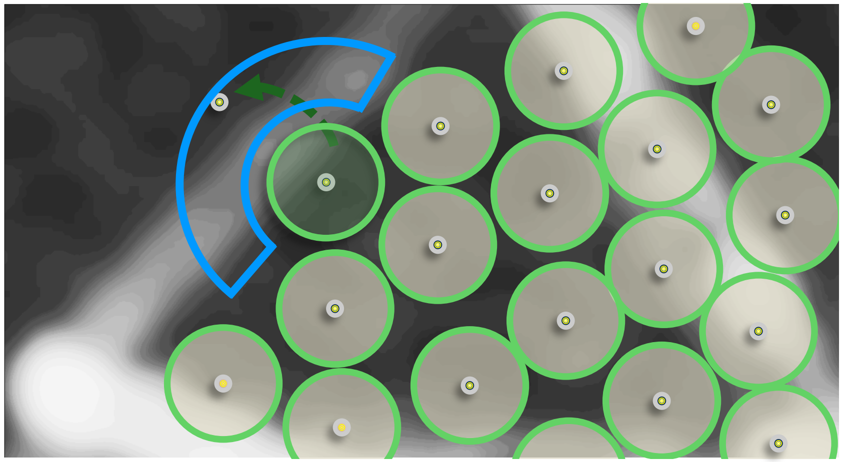


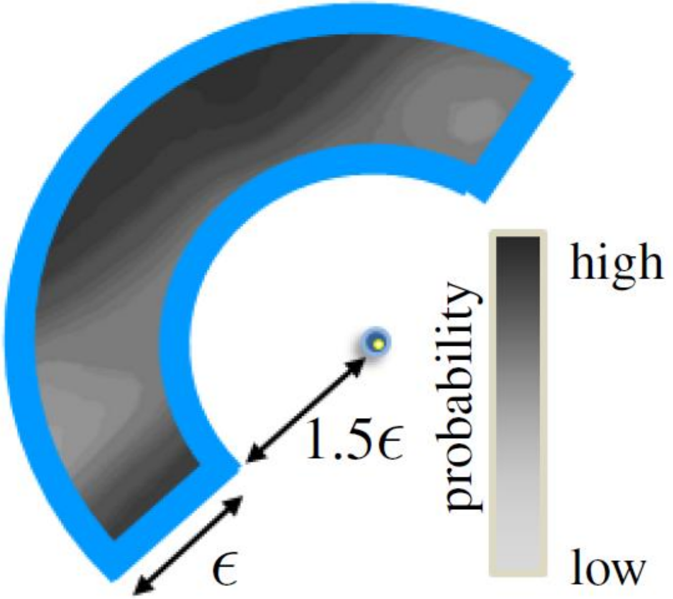
Poisson disk sampling guided by image gradient

Step 3: homogeneization

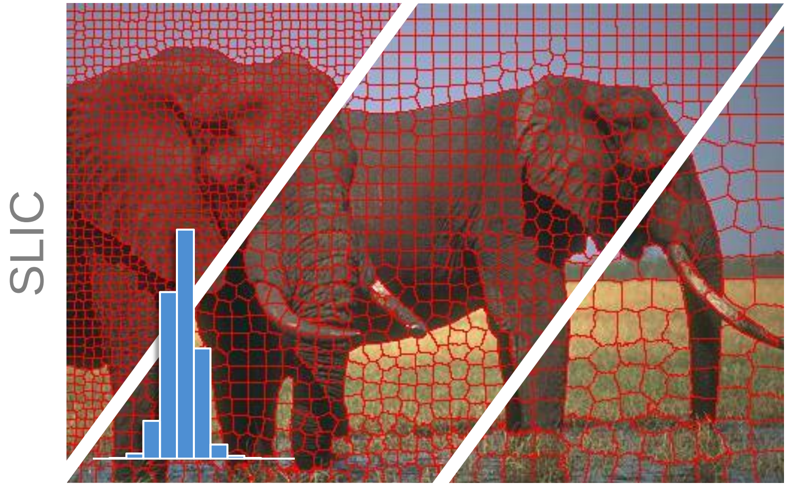


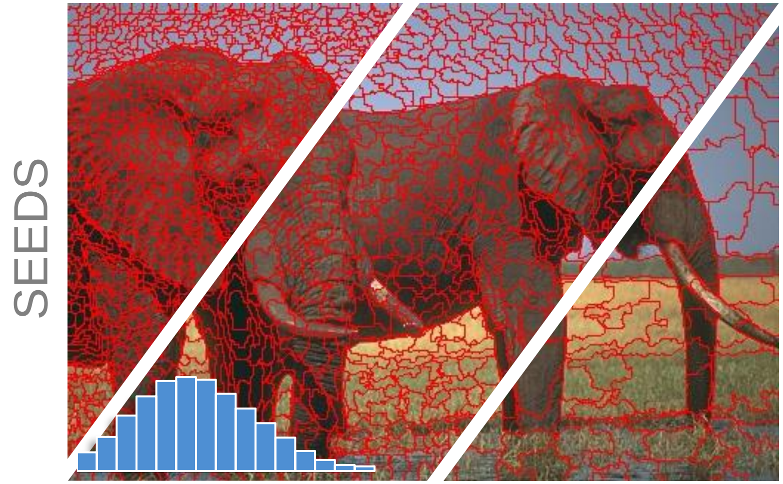


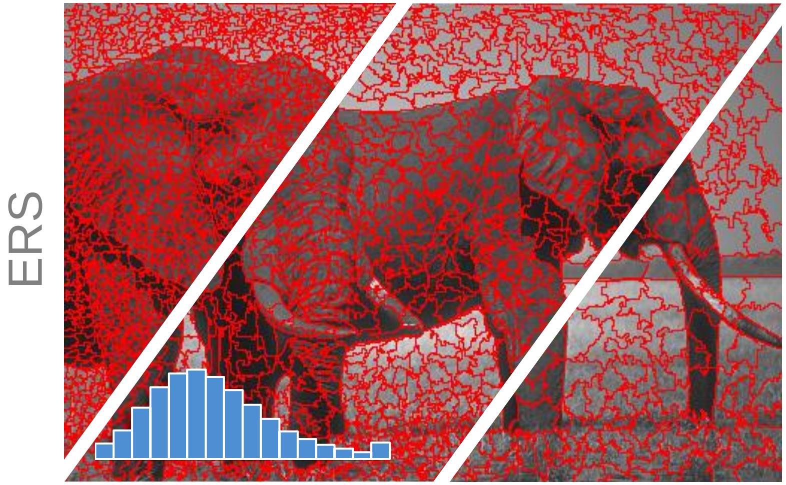


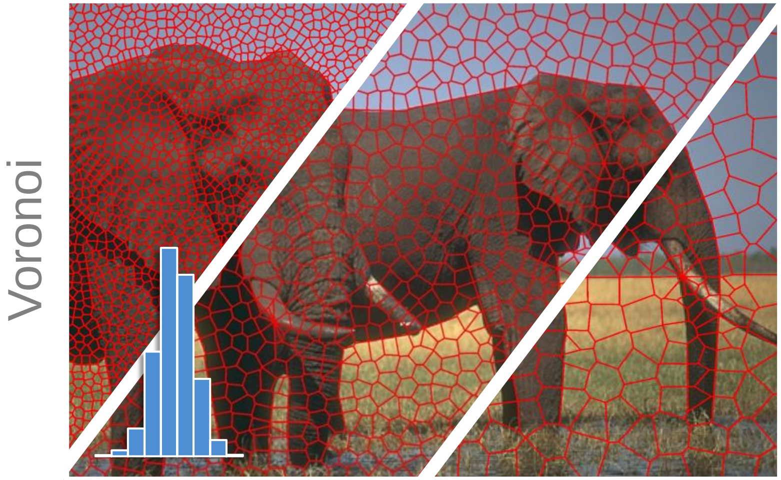


Comparisons with superpixel methods



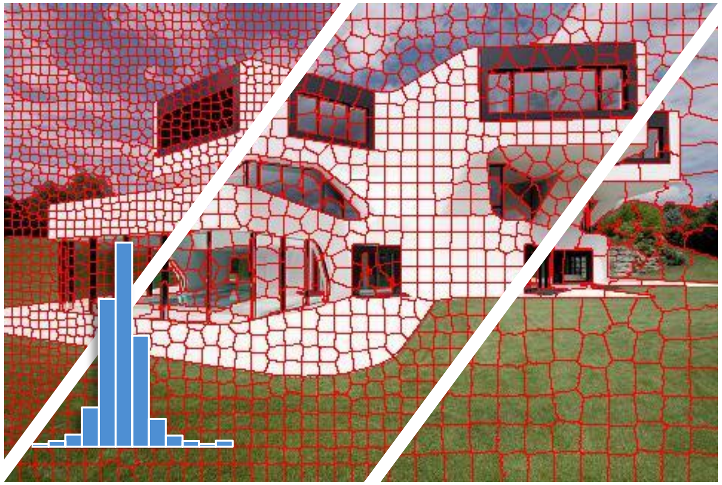




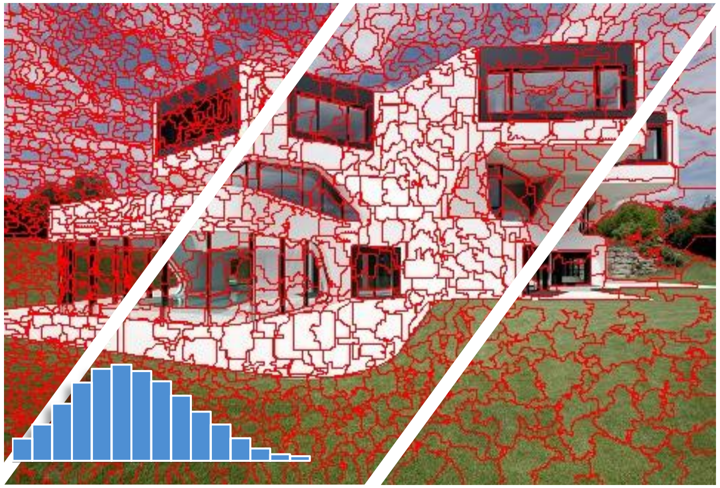


Comparisons with superpixel methods

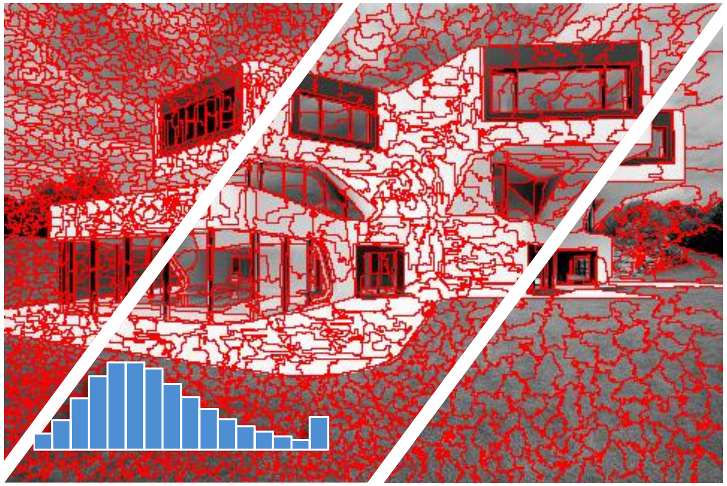
SLIC



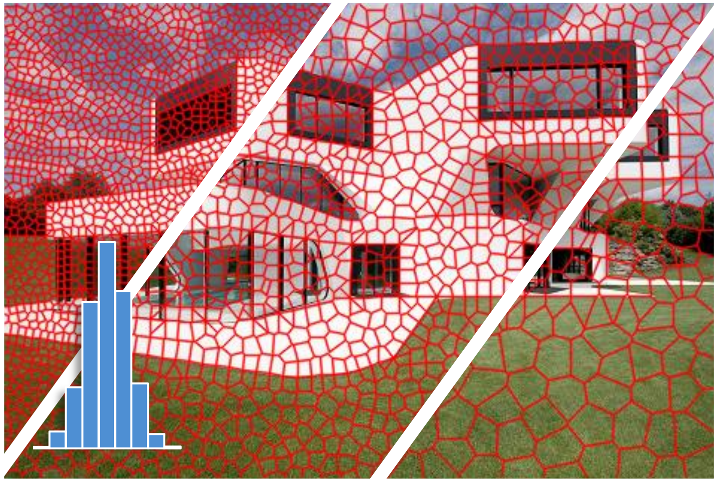
SEEDS



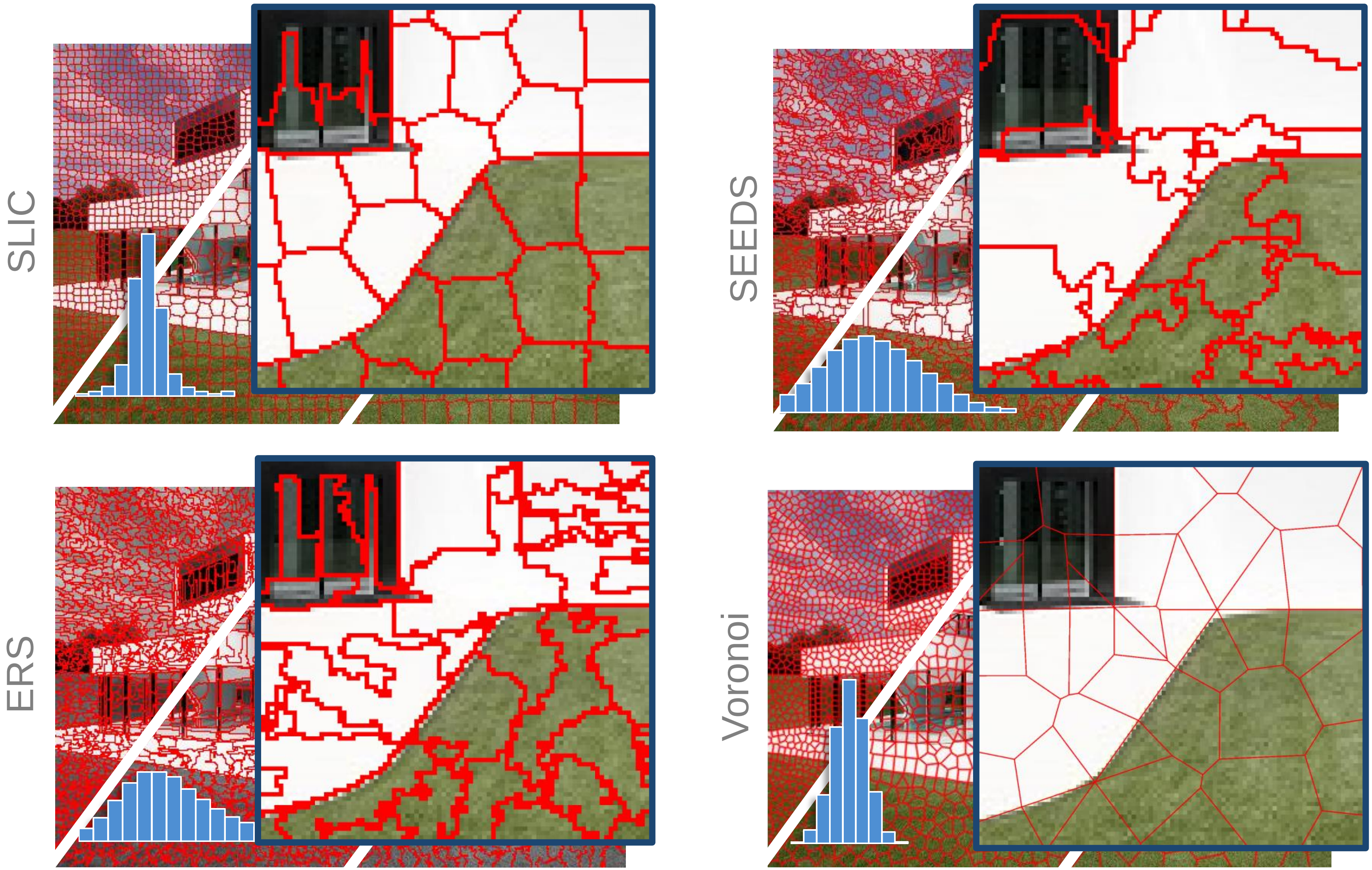
ERS



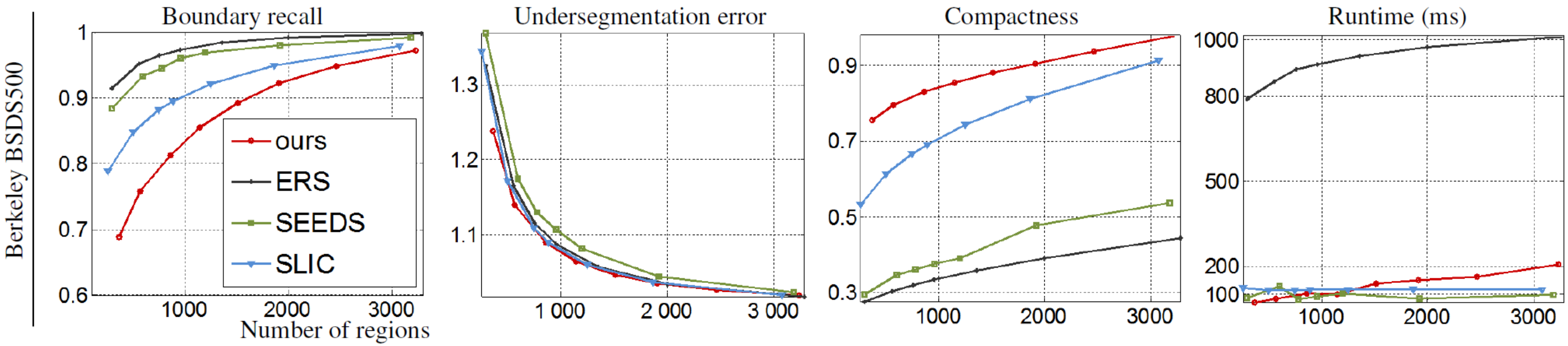
Voronoi



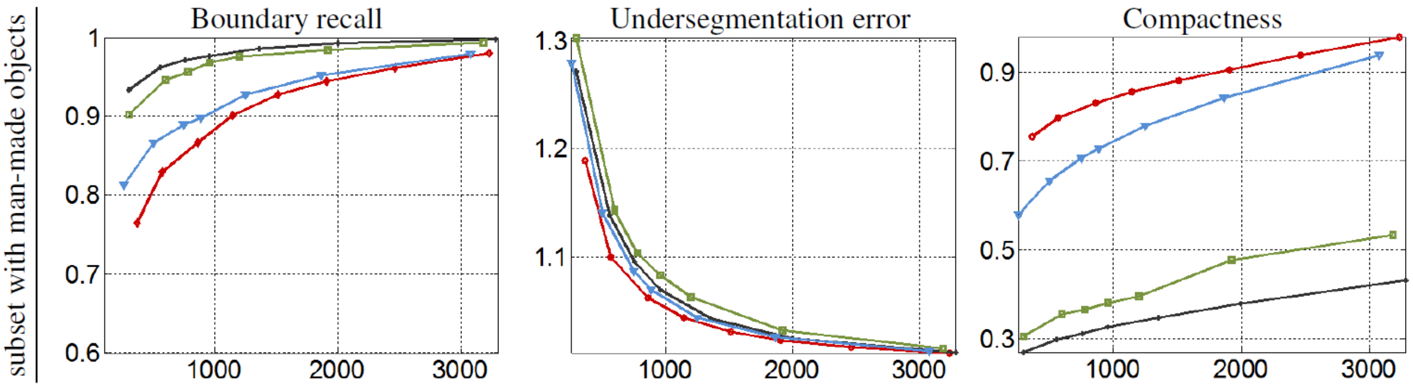
Comparisons with superpixel methods



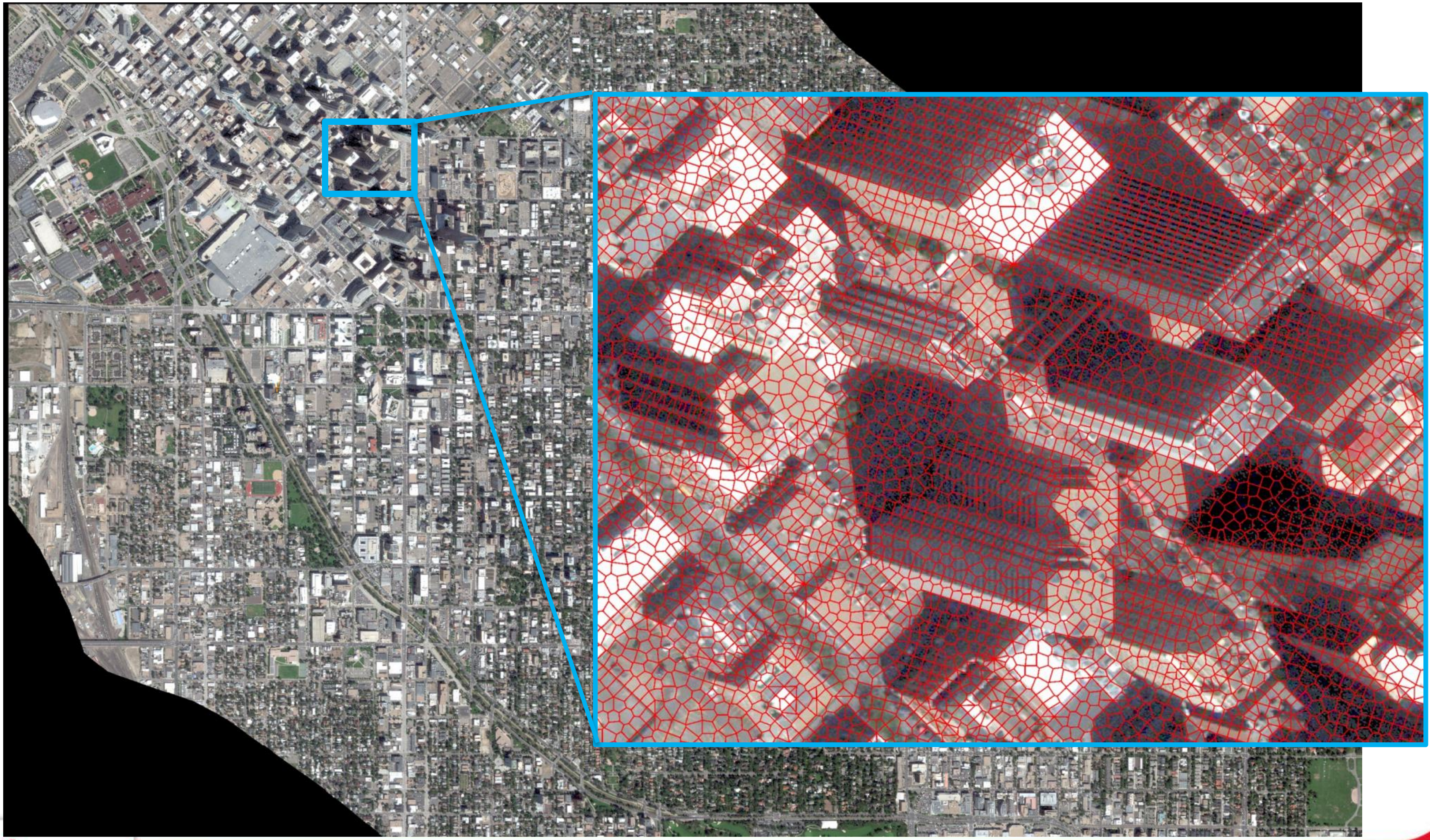
Comparisons with superpixel methods



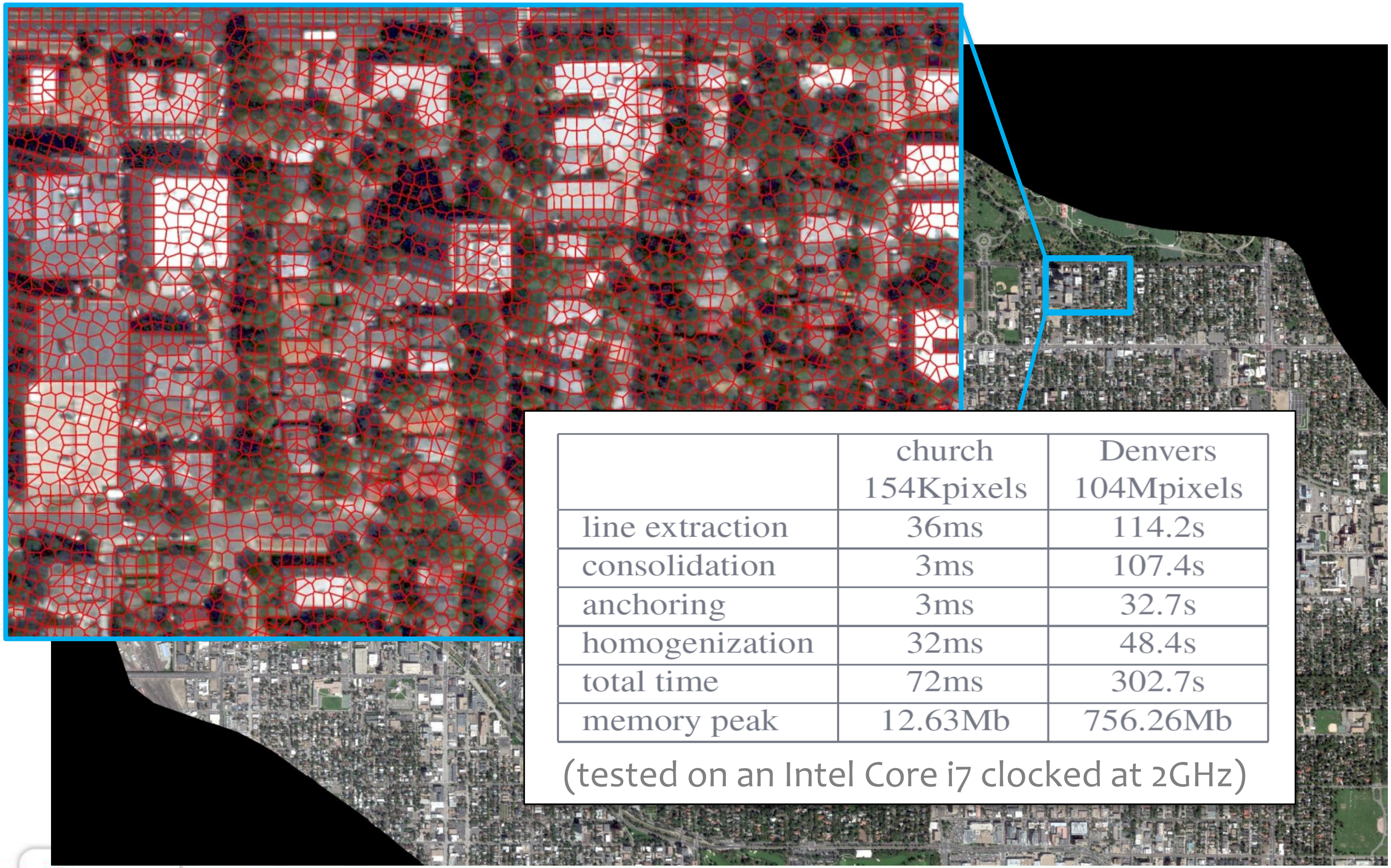
Comparisons with superpixel methods



Results on very big images



Results on very big images



	church 154Kpixels	Denvers 104Mpixels
line extraction	36ms	114.2s
consolidation	3ms	107.4s
anchoring	3ms	32.7s
homogenization	32ms	48.4s
total time	72ms	302.7s
memory peak	12.63Mb	756.26Mb

(tested on an Intel Core i7 clocked at 2GHz)

Demo

One parameter

ε : expected width of the Voronoi cells

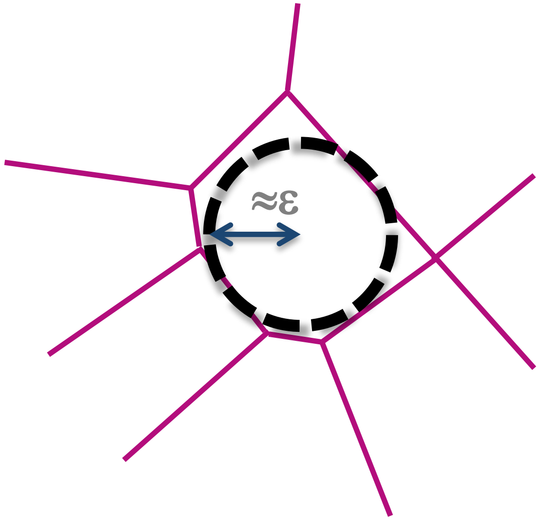
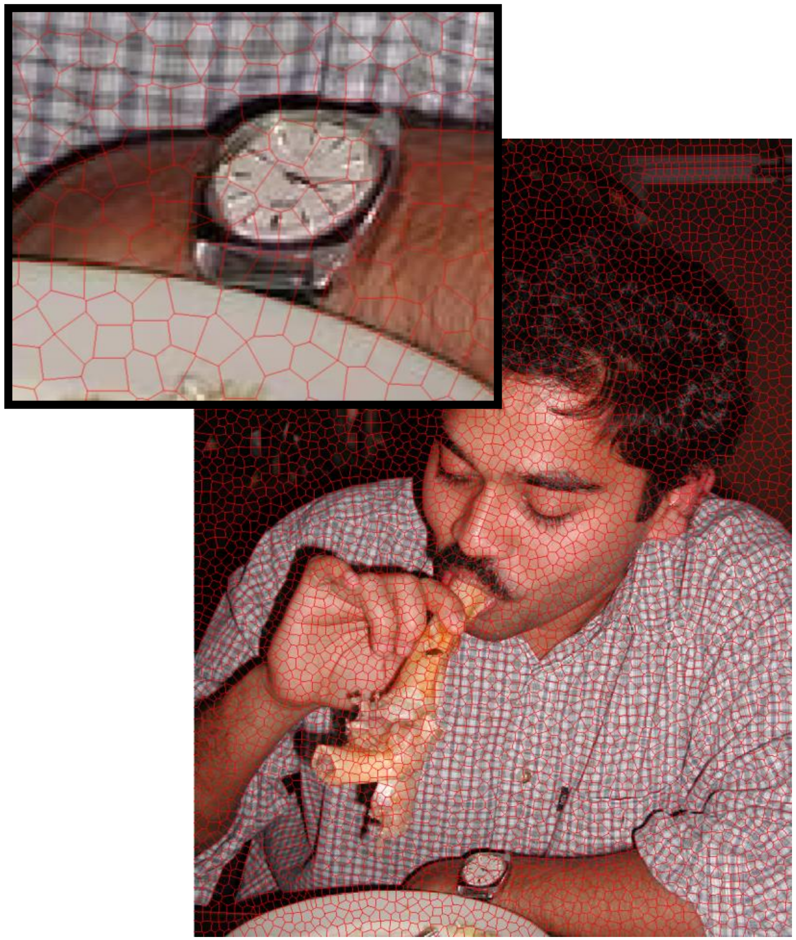


Image partitioning with a kinetic data-structure

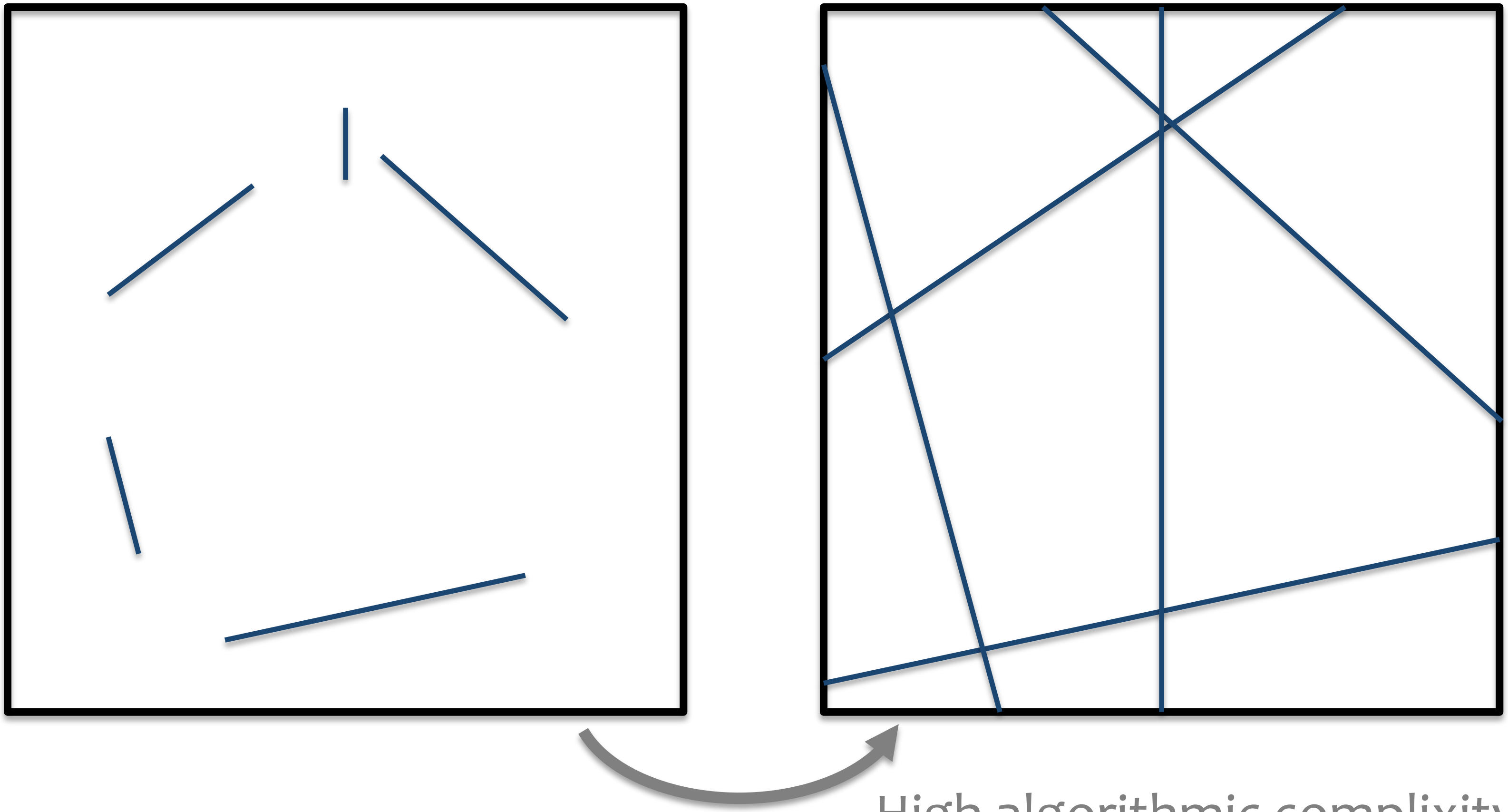
[Bauchet and Lafarge, KIPPI: Kinetic Polygonal Partitioning of Images, CVPR 2018]

Cells with heterogeneous size



Less cells but
more meaningful

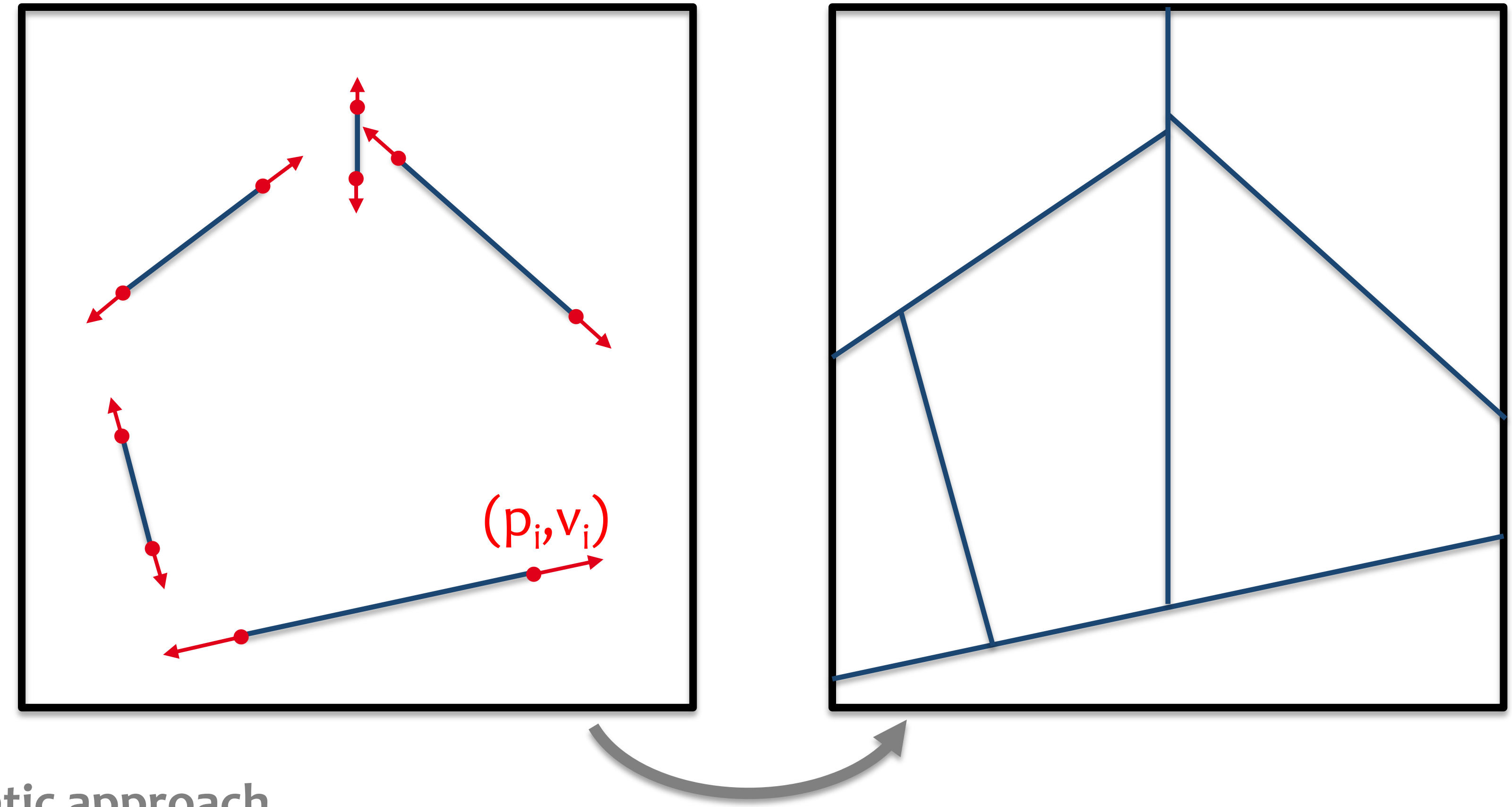
Polygonal partitioning as space cutting



High algorithmic complixity

Generate many cells

Polygonal partitioning as space cutting



Kinetic approach

Shape intersection not greedy anymore, but based on shape proximity

kinetic formulation

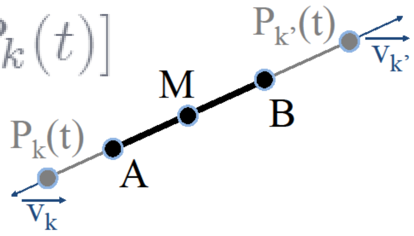
Kinetic data-structure: a dynamic planar graph $G_t = (V_t, E_t)$

kinetic formulation

Kinetic data-structure: a dynamic planar graph $G_t = (V_t, E_t)$

Primitive: a dynamic segment $s_k(t) = [M P_k(t)]$

with $P_k(t) = A + \overrightarrow{v_k} \times t$

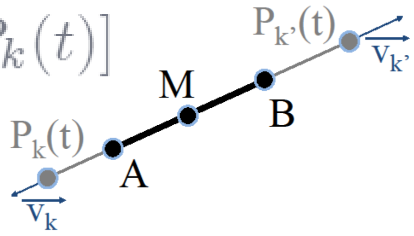


kinetic formulation

Kinetic data-structure: a dynamic planar graph $G_t = (V_t, E_t)$

Primitive: a dynamic segment $s_k(t) = [M P_k(t)]$

with $P_k(t) = A + \overrightarrow{v_k} \times t$



Certificate

Function testing the intersection of primitive i with other primitives at time t

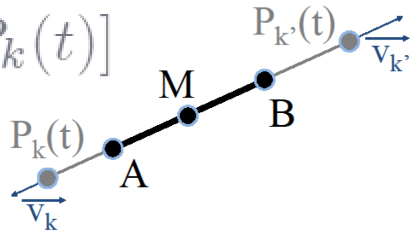
$$C_i(t) = \prod_{\substack{j=1 \\ j \neq i}}^{N} Pr_{i,j}(t) \quad \text{with} \quad Pr_{i,j}(t) = \begin{cases} 1 & \text{if } d(P_i(t), s_j(t)) > 0 \\ 0 & \text{otherwise} \end{cases}$$

kinetic formulation

Kinetic data-structure: a dynamic planar graph $G_t = (V_t, E_t)$

Primitive: a dynamic segment $s_k(t) = [M P_k(t)]$

with $P_k(t) = A + \overrightarrow{v_k} \times t$



Certificate

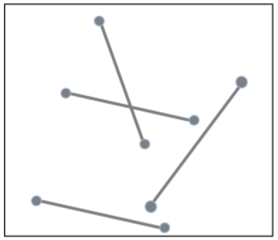
Function testing the intersection of primitive i with other primitives at time t

$$C_i(t) = \prod_{\substack{j=1 \\ j\neq i}}^{N} Pr_{i,j}(t) \quad \text{with} \quad Pr_{i,j}(t) = \begin{cases} 1 & \text{if } d(P_i(t), s_j(t)) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Queue of events

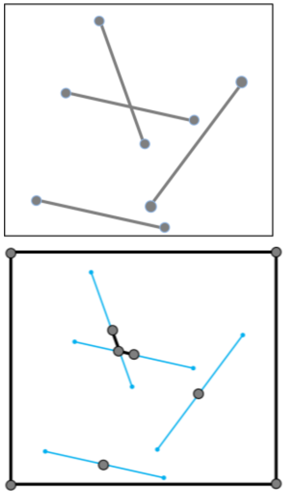
List of times t indicating when a certificate is equal to 0
(ranked by ascending order)

kinetic formulation



Algorithm

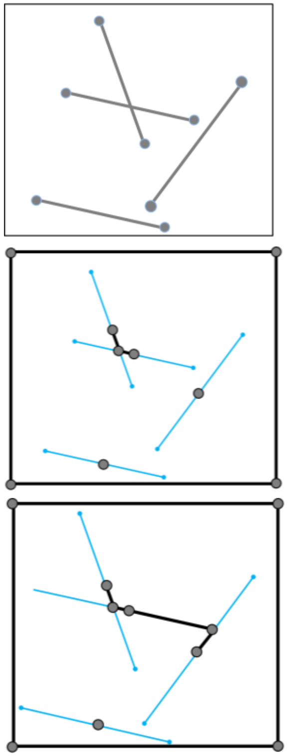
kinetic formulation



Algorithm

- Initialize the data-structure by inserting points where two line-segments intersect

kinetic formulation



Algorithm

- Initialize the data-structure by inserting points where two line-segments intersect

- For each event of the queue,

    update the data-structure

    test the deactivation of the primitive

    update the queue of events

kinetic formulation

Algorithm

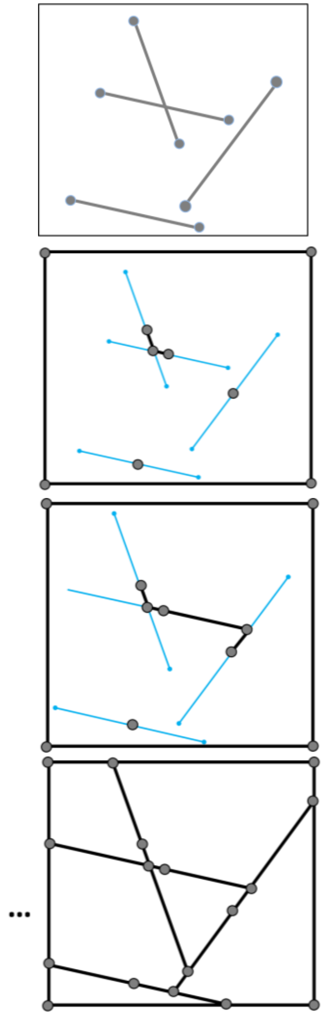
- Initialize the data-structure by inserting points where two line-segments intersect

- For each event of the queue,

  update the data-structure

  test the deactivation of the primitive

  update the queue of events



kinetic formulation

Algorithm

- Initialize the data-structure by inserting points where two line-segments intersect

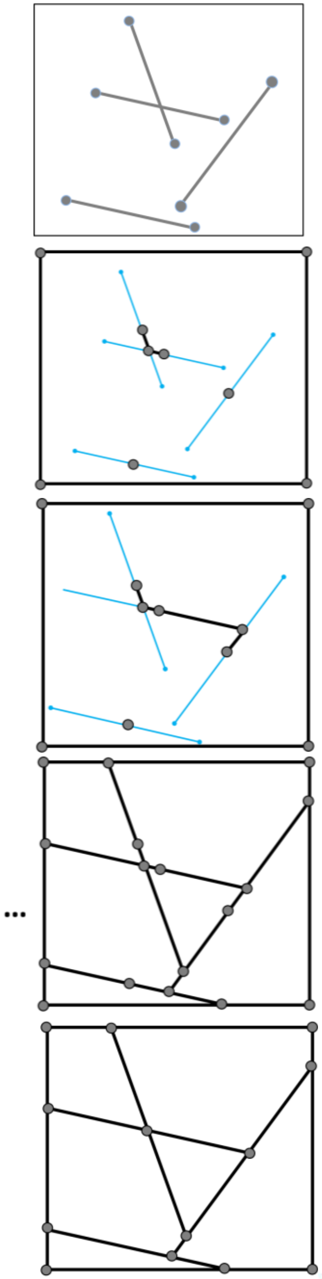
- For each event of the queue,

  update the data-structure

  test the deactivation of the primitive

  update the queue of events

- Finalization



kinetic formulation

Flexibility

- Use the confidence for each detected line-segments to better adapt the partition (increase speed of good line-segments)

kinetic formulation

Flexibility

- Use the confidence for each detected line-segments to better adapt the partition (increase speed of good line-segments)

- Policy for deactivating a primitive
  - Impose a maximal number of intersection K per primitive
  - Check the alignment of a potential prolongation with image gradients

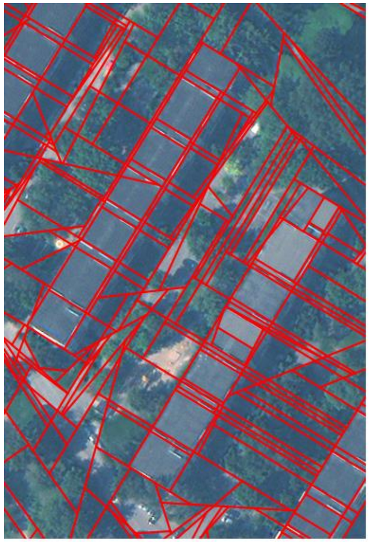
kinetic formulation

Flexibility

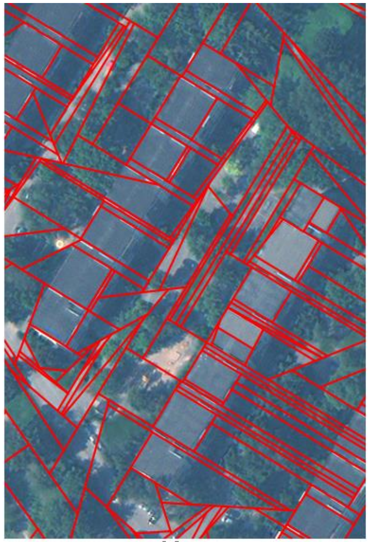
- Use the confidence for each detected line-segments to better adapt the partition (increase speed of good line-segments)

- Policy for deactivating a primitive

- Impose a maximal number of intersection K per primitive
- Check the alignment of a potential prolongation with image gradients



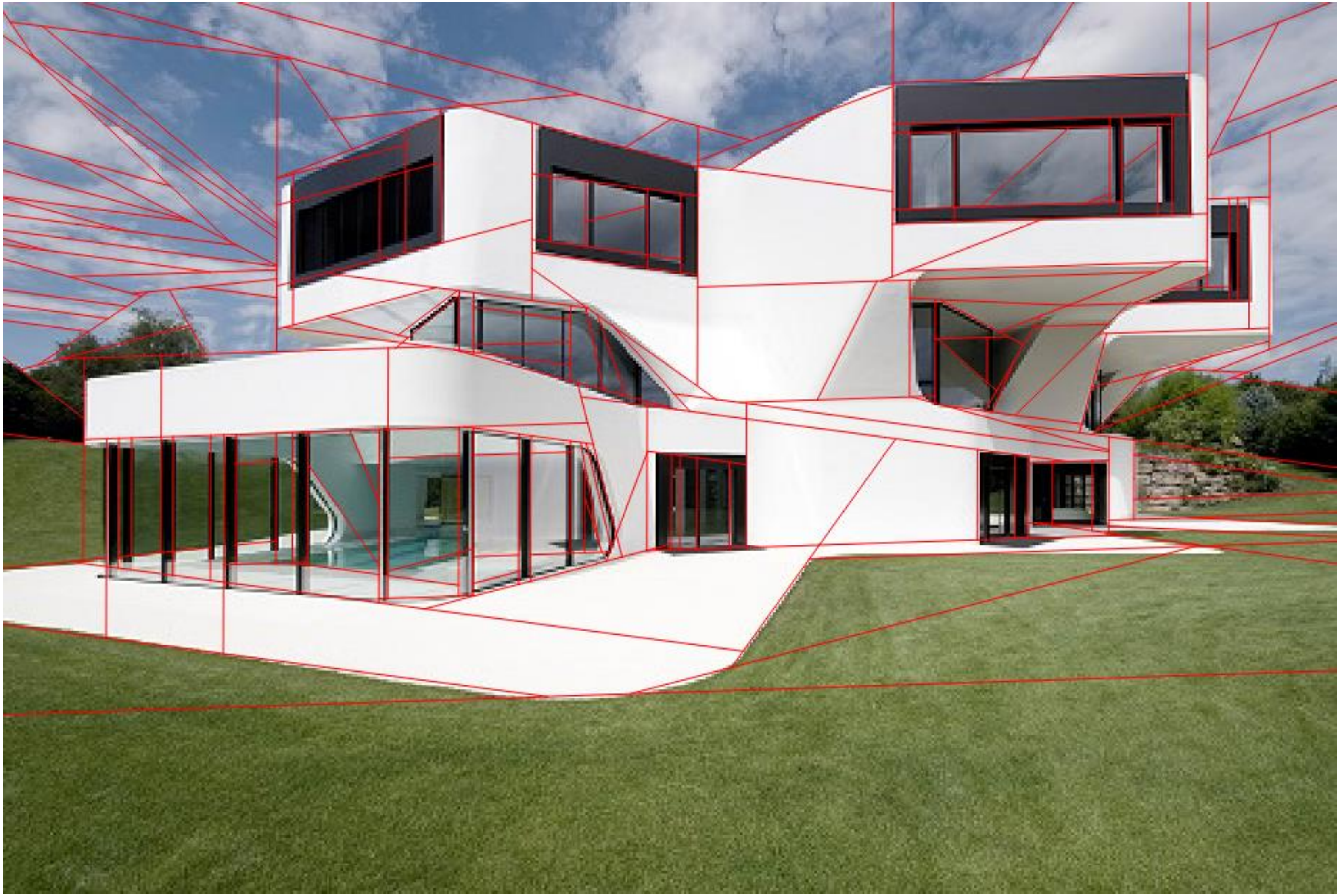
K=1 + gradients



K=1
(motorcycle graph)

K=20

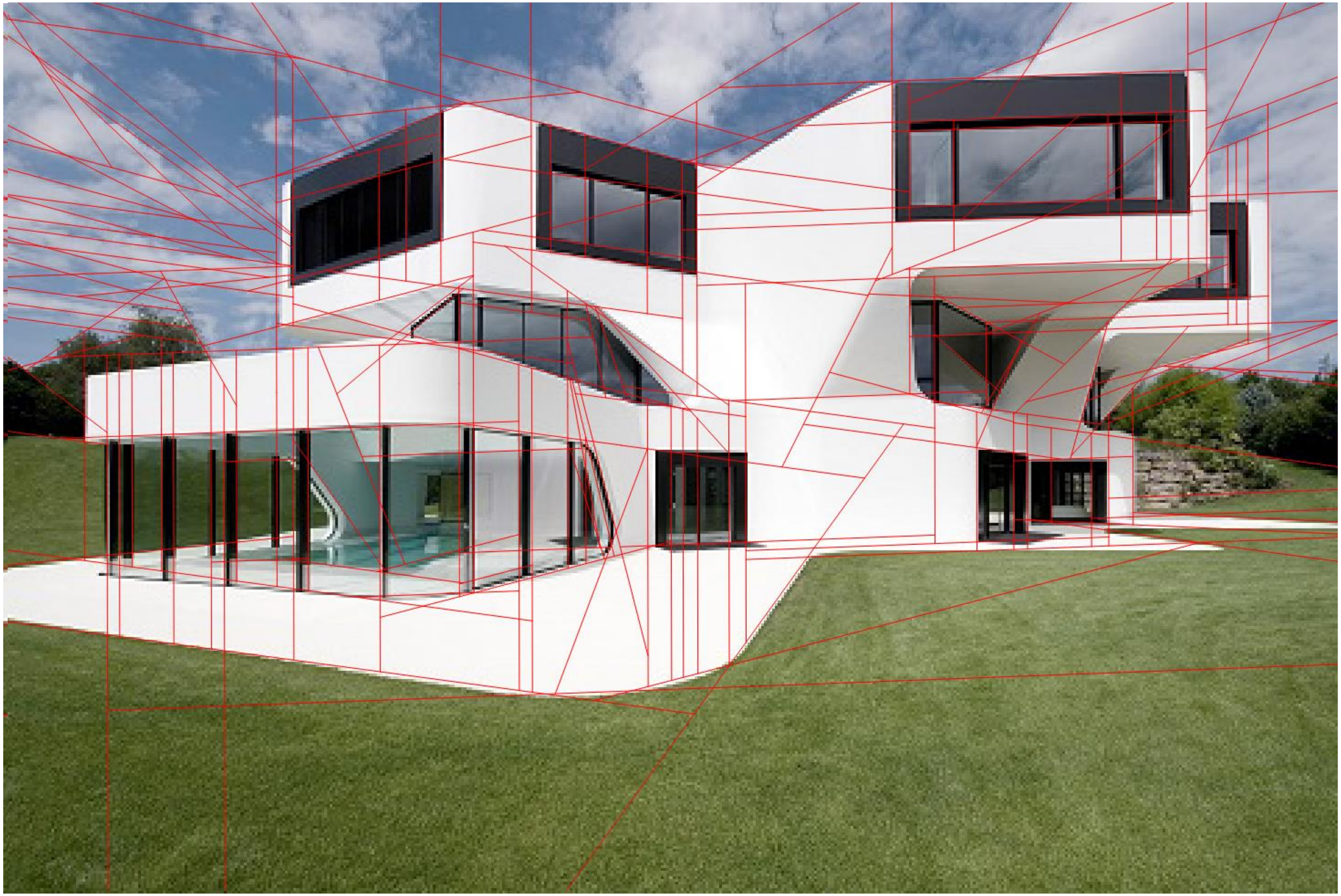
kinetic formulation



K=1

226 cells

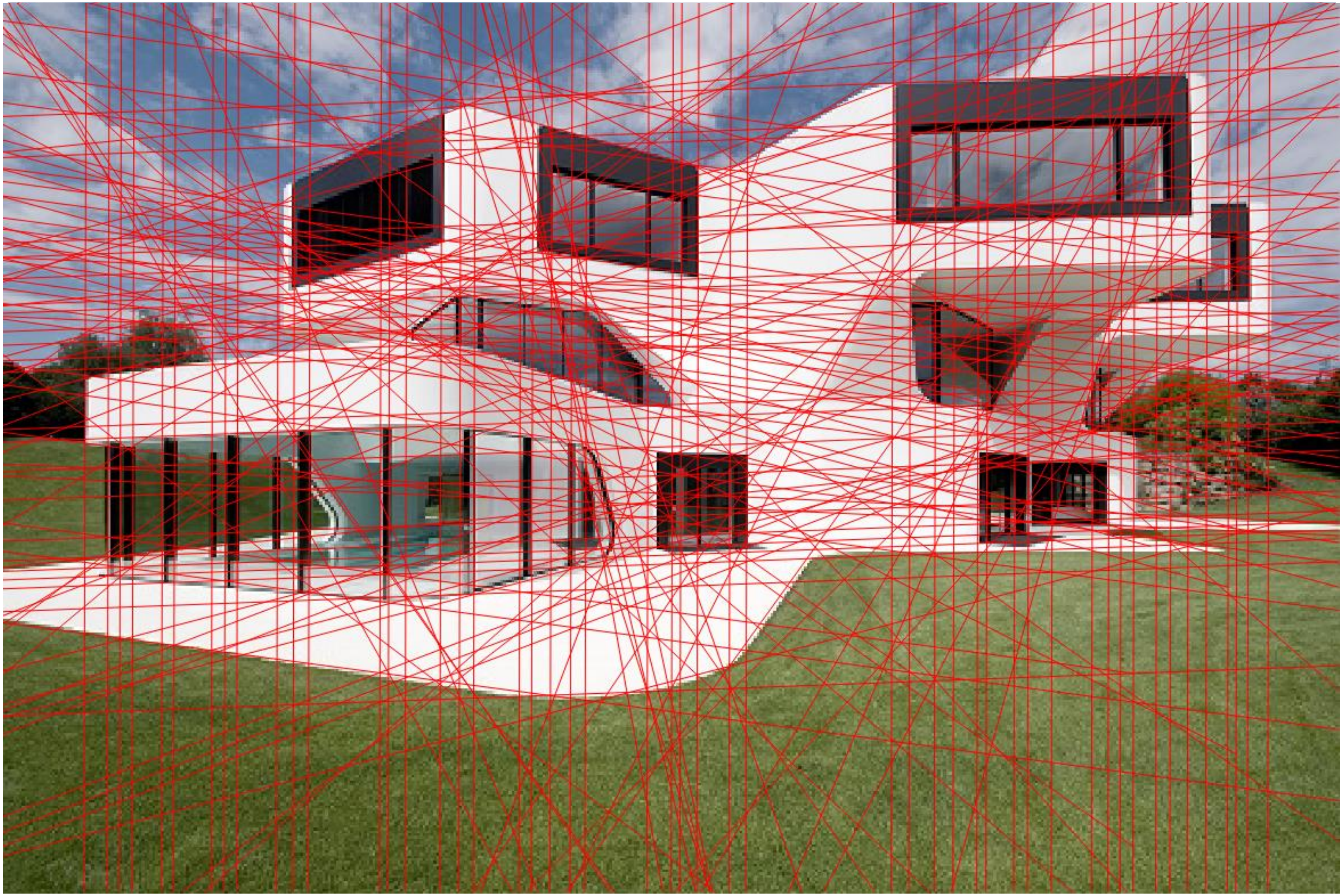
kinetic formulation



K=2

593 cells

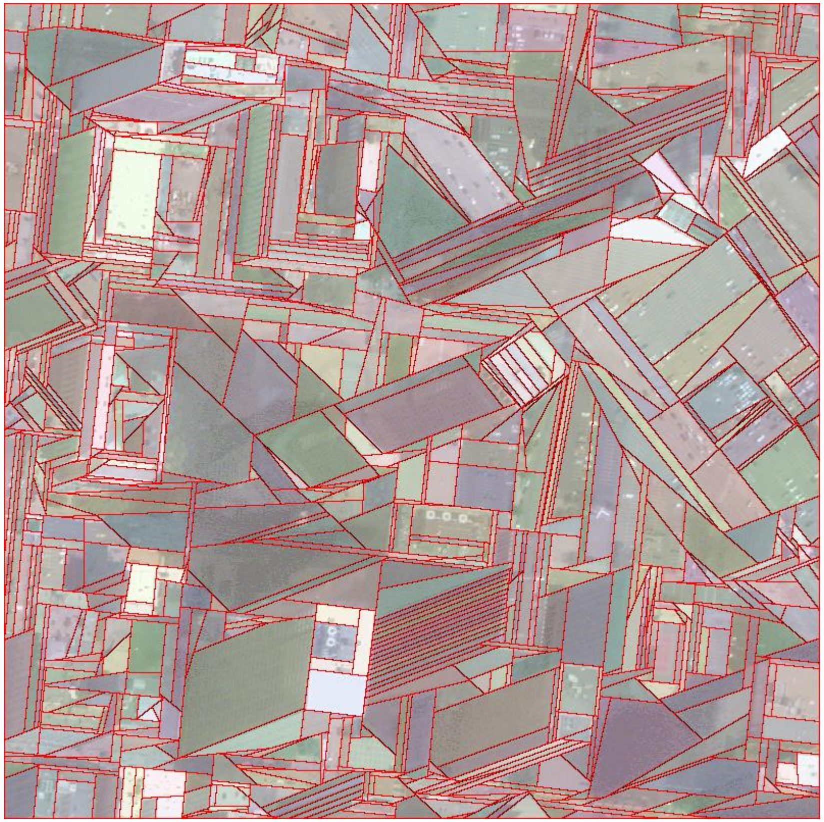
kinetic formulation



K= ∞

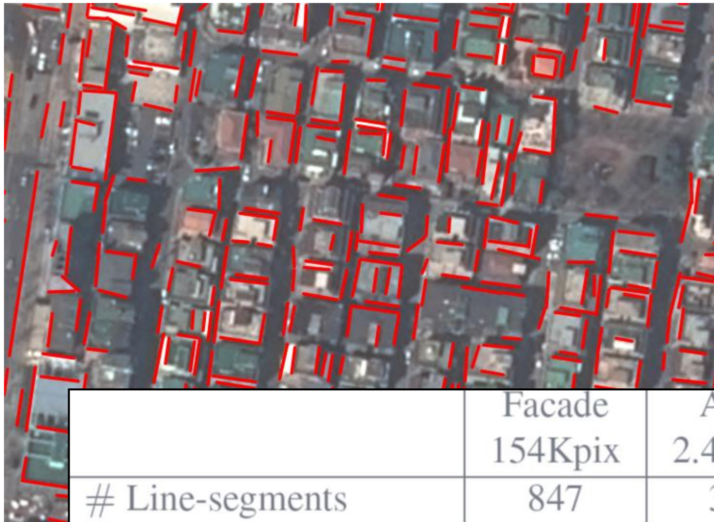
9.8k cells

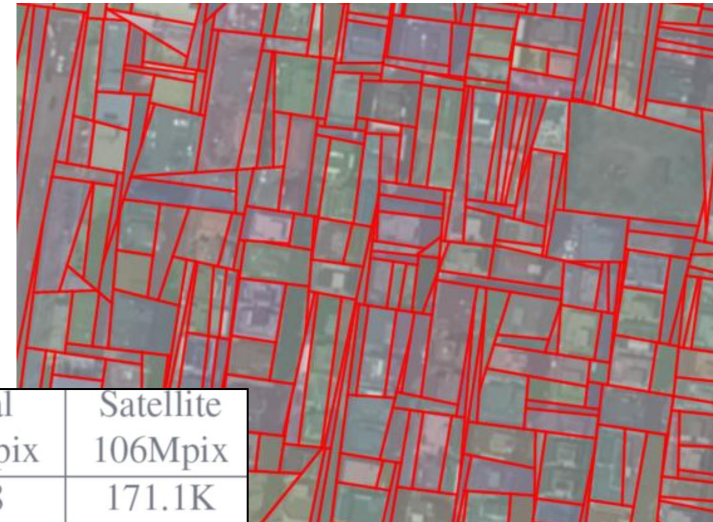
Results on satellite images



Results on satellite images

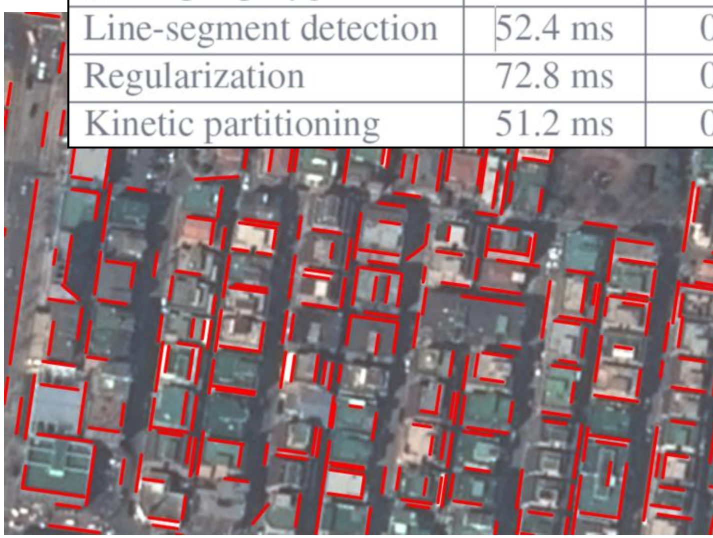
Without line-segment regularization

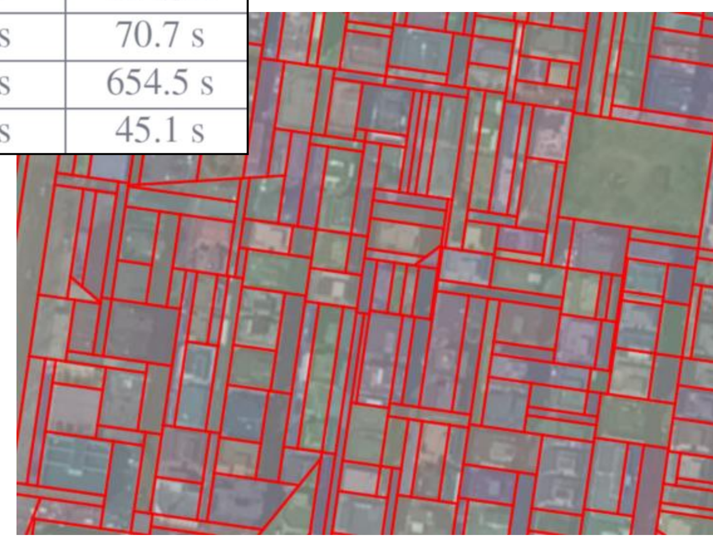




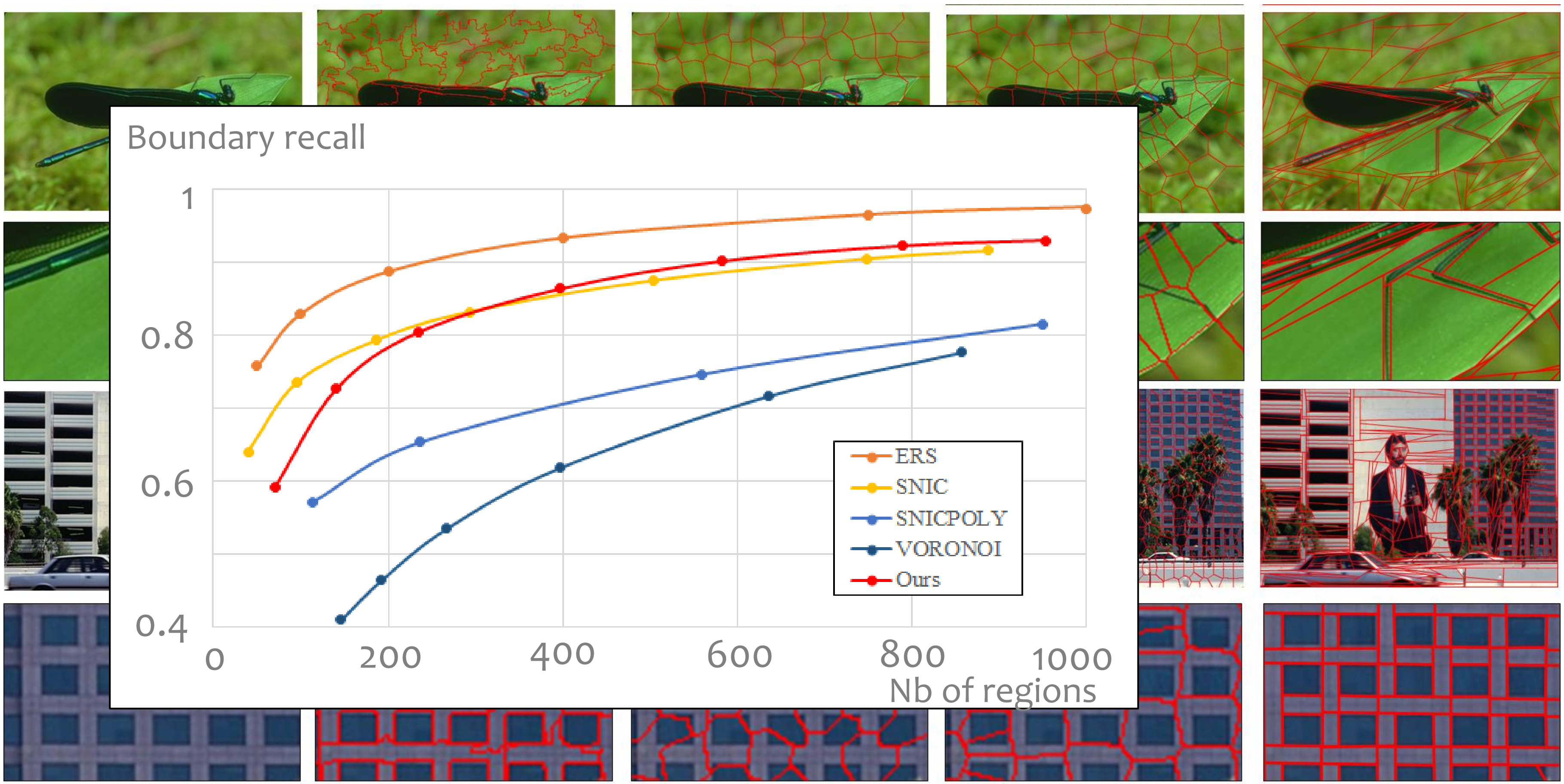
	Facade 154Kpix	Aerial 2.46Mpix	Satellite 106Mpix
# Line-segments	847	3178	171.1K
# Output polygons	530	2488	124.5K
Line-segment detection	52.4 ms	0.59 s	70.7 s
Regularization	72.8 ms	0.35 s	654.5 s
Kinetic partitioning	51.2 ms	0.23 s	45.1 s

With line-segment regularization



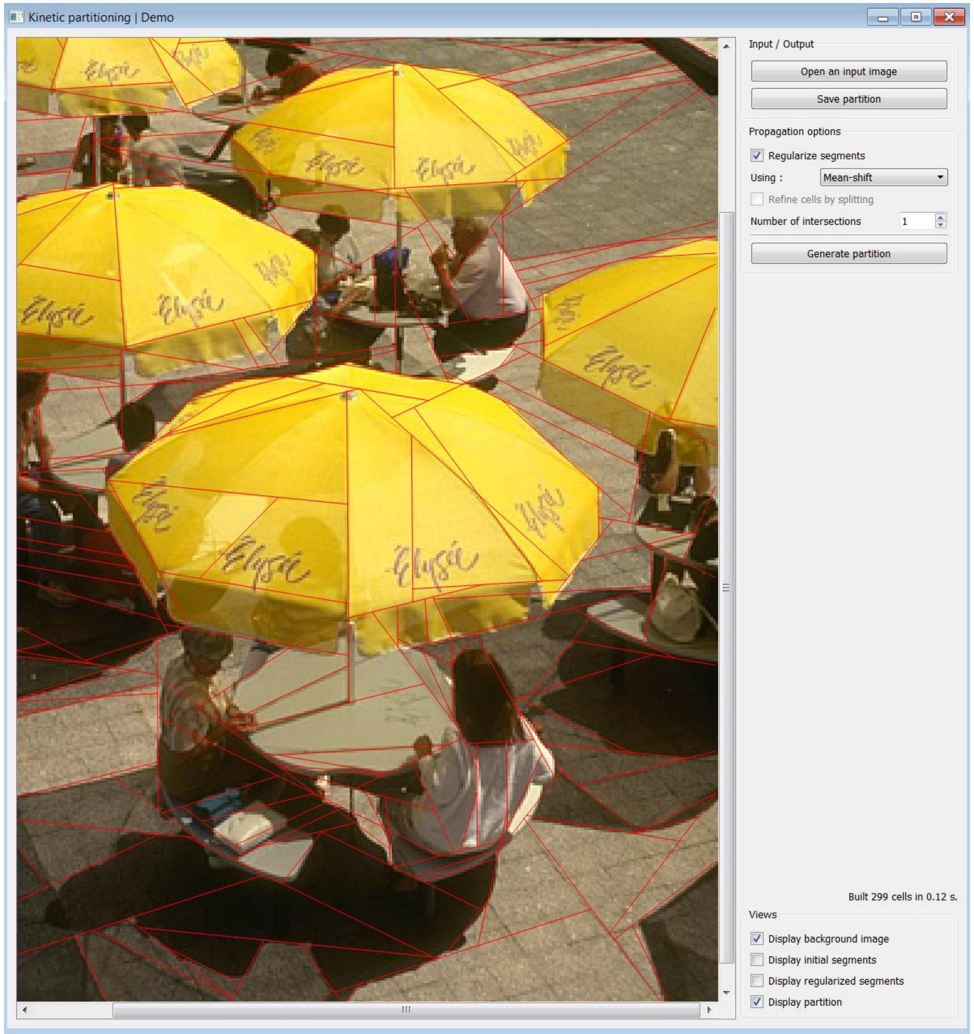


Comparisons with over-segmentation methods



ERS SNICPOLY VORONOI ours

Demo



Kinetic partitioning | Demo
Input / Output
Open an input image
Save partition
Propagation options
Regularize segments
Using :
Mean-shift
Refine cells by splitting
Number of intersections
1
Generate partition
Built 299 cells in 0.12 s.
Views
Display background image
Display initial segments
Display regularized segments
Display partition

Application to object contouring

Object contouring

Label each cell as inside or outside the objects of interest

Object contouring

Label each cell as inside or outside the objects of interest

Graph-cut

Data term: distance to a saliency map

$$H(i|m_f) = \frac{\min_{j \in S_{m_f}} \|I(i) - \widehat{I}(j)\|_2^2}{\min_{j \in S_0} \|I(i) - \widehat{I}(j)\|_2^2 + \min_{j \in S_1} \|I(i) - \widehat{I}(j)\|_2^2}$$

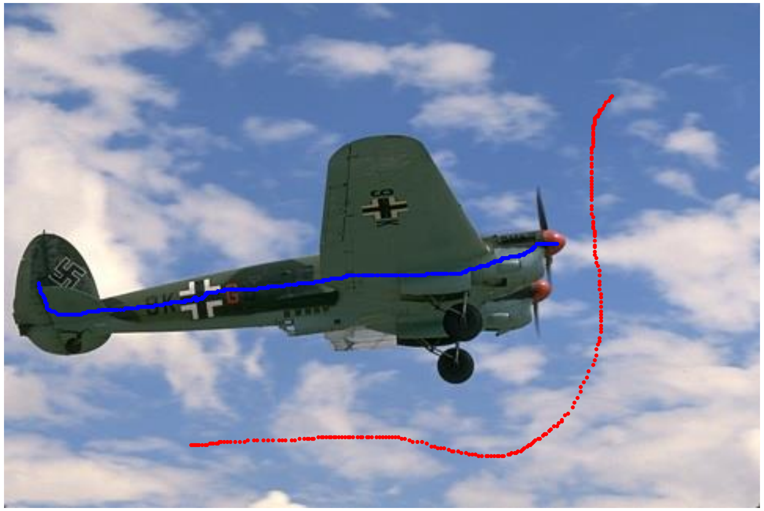
Object contouring

Label each cell as inside or outside the objects of interest

Graph-cut

- Data term: distance to a saliency map
- Potential: Potts model

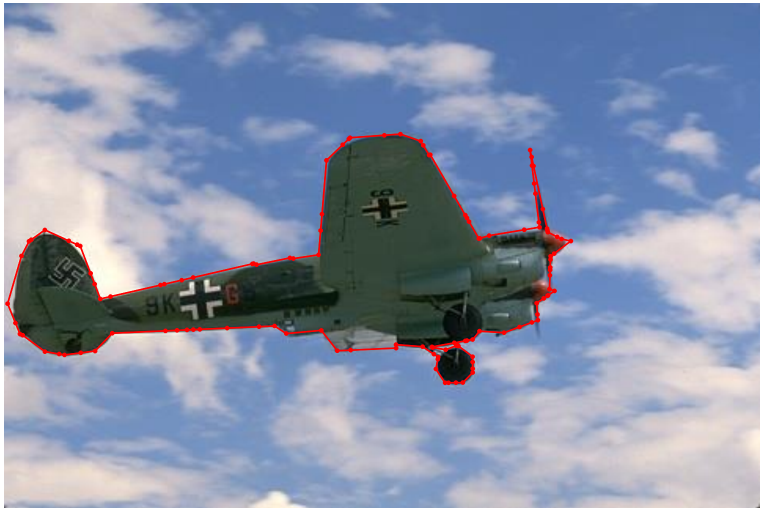
Object contouring



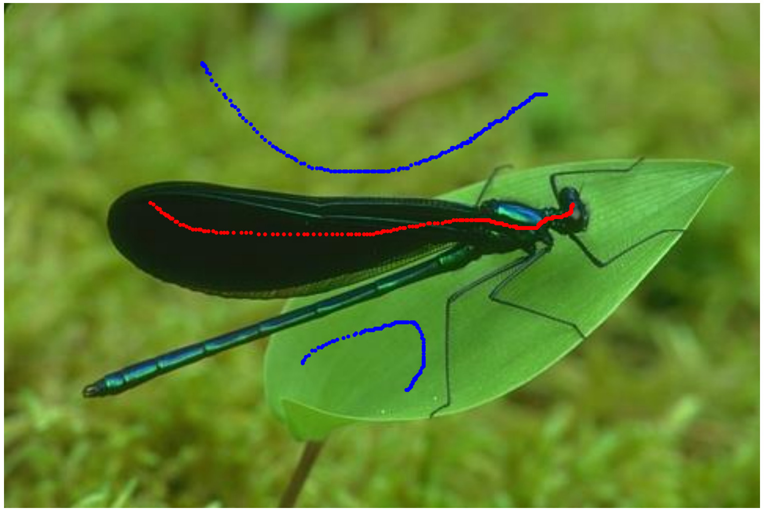
with Voronoi

Grabcut + vectorization

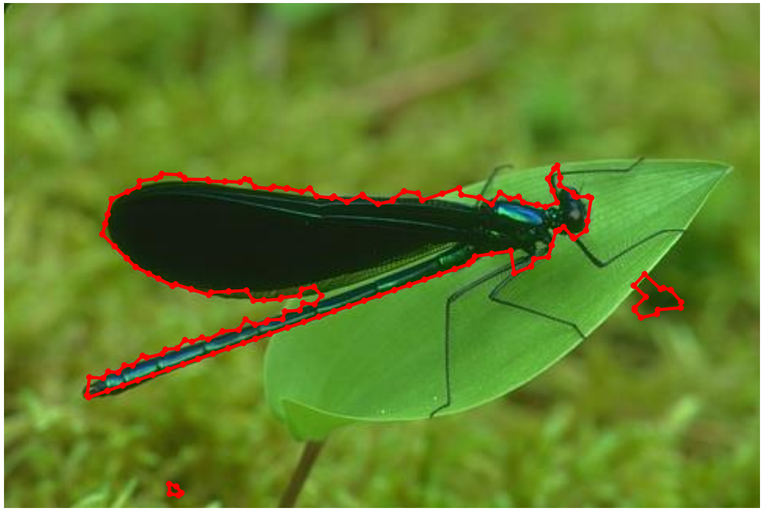
ours



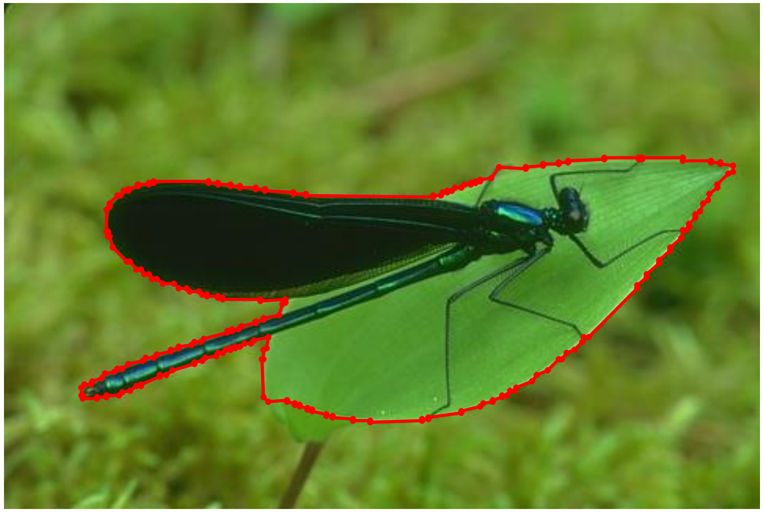
Object contouring



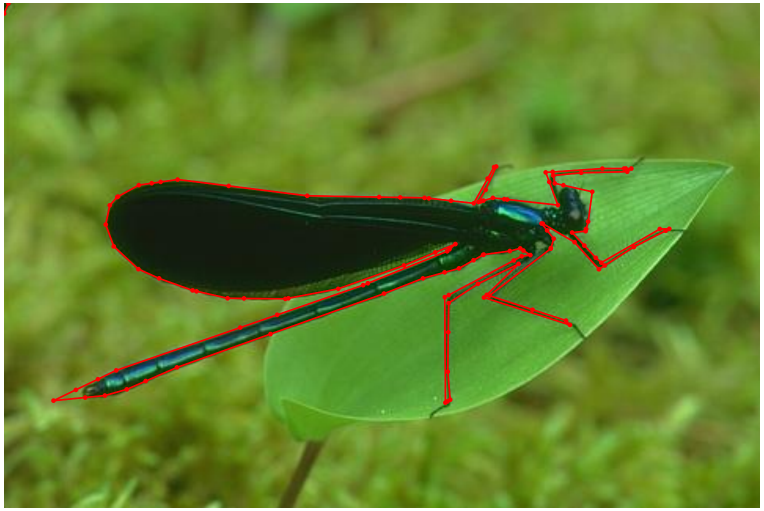
with Voronoi



Grabcut + vectorization

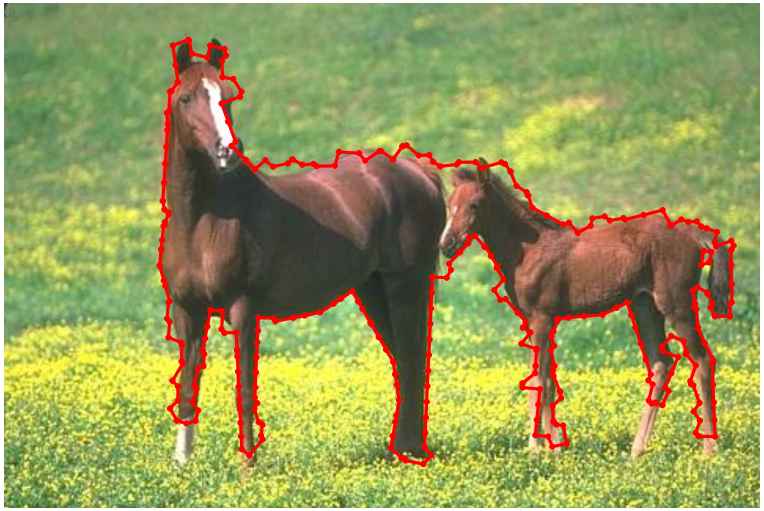


ours



Object contouring

with Voronoi



Grabcut + vectorization

ours

Object contouring

with Voronoi

Grabcut + vectorization

ours

Object contouring

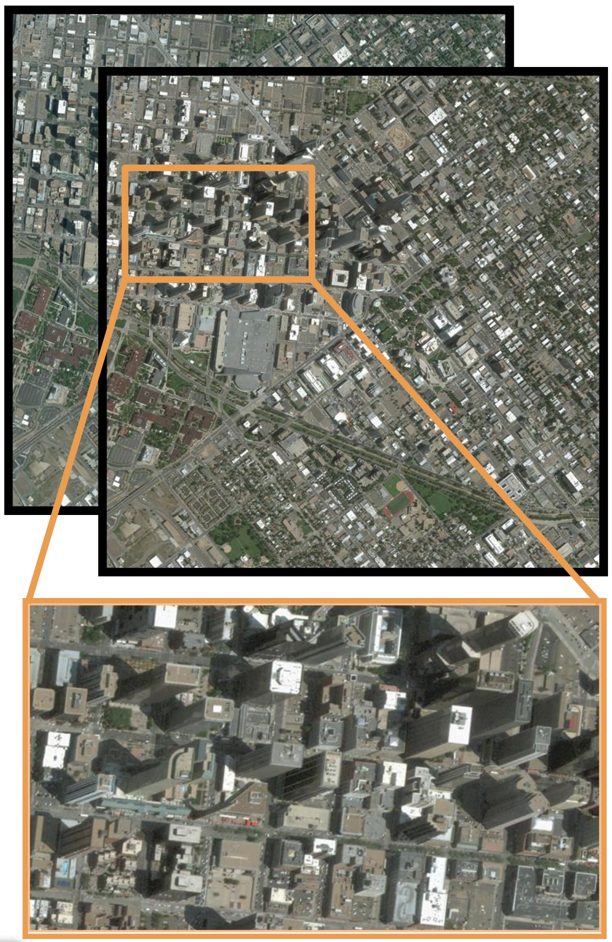
Nb of edges: 130 | Nb of edges: 308 | Nb of edges: 476

Application to city modeling from satellite images

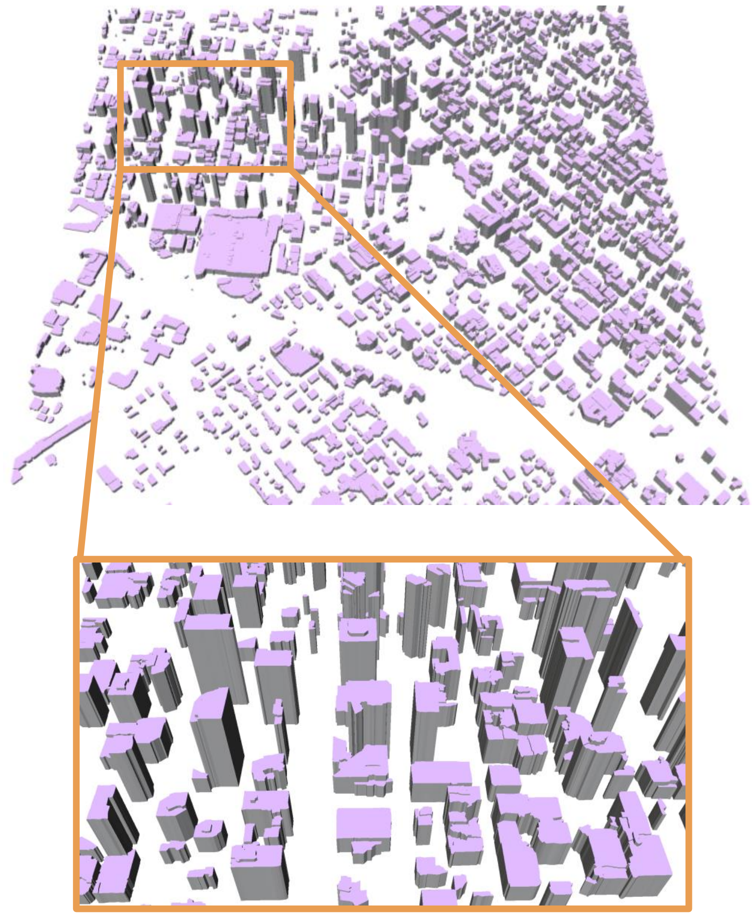
[Duan and Lafarge, Towards large-scale city reconstruction from satellites, ECCV 2016]

City modeling from satellite images

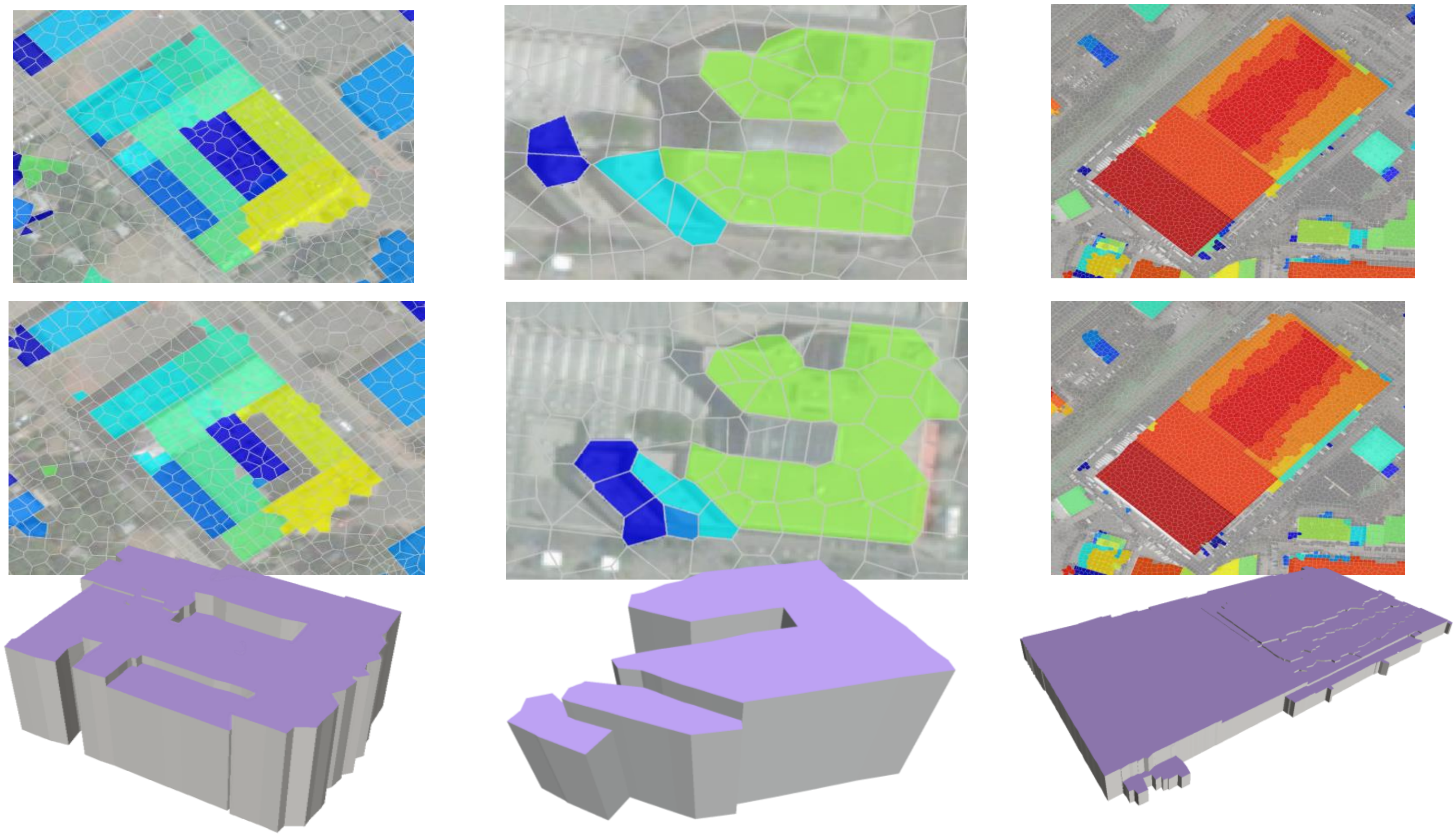
Input
Stereo pair of satellite images



Output
3D model at LOD1



City modeling from satellite images

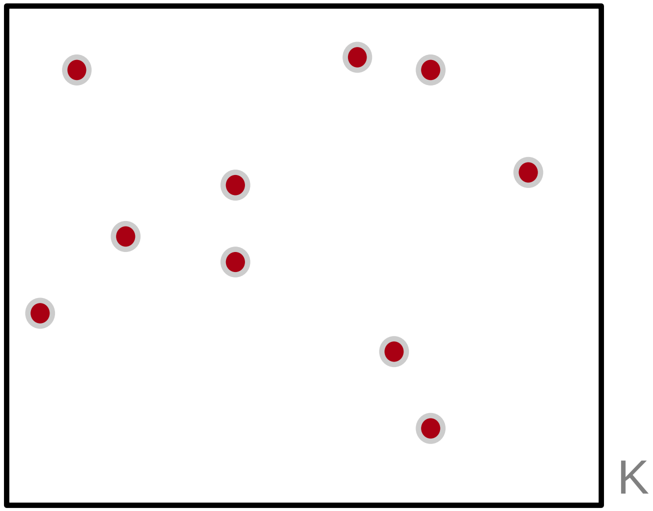


2

Delaunay point processes

Background on point process

Random configurations of points distributed in a bounded domain K

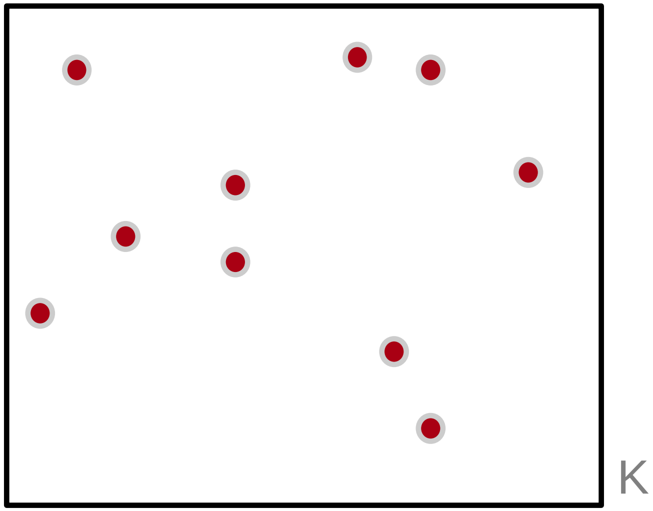


Background on point process

Random configurations of points distributed in a bounded domain K

Interesting characteristics

- **#points is a random variable**

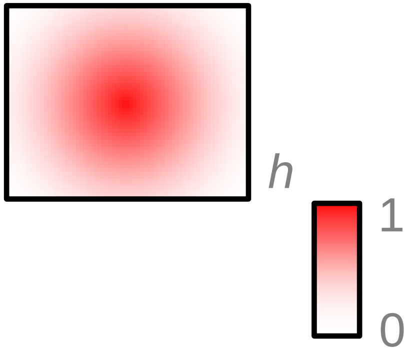


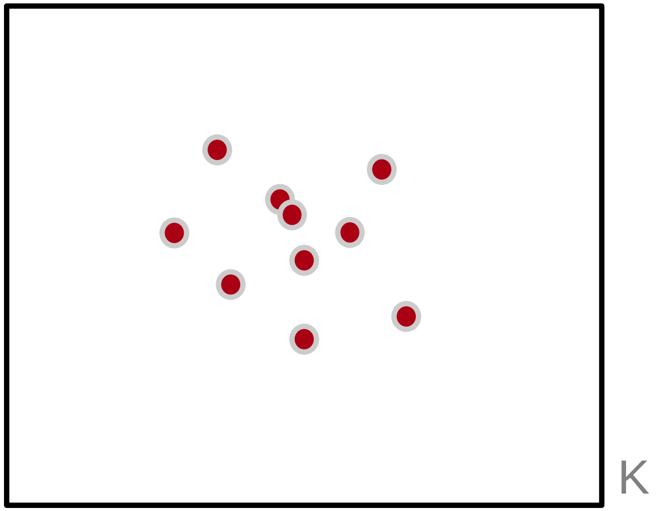
Background on point process

Random configurations of points distributed in a bounded domain K

Interesting characteristics

- #points is a random variable
- **can be guided by a density *h***



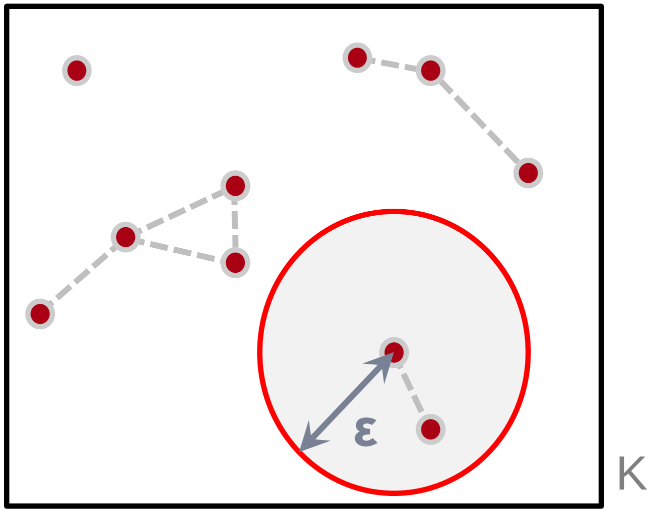


Background on point process

Random configurations of points distributed in a bounded domain K

Interesting characteristics

- #points is a random variable
- can be guided by a density *h*
- **with spatial interactions**

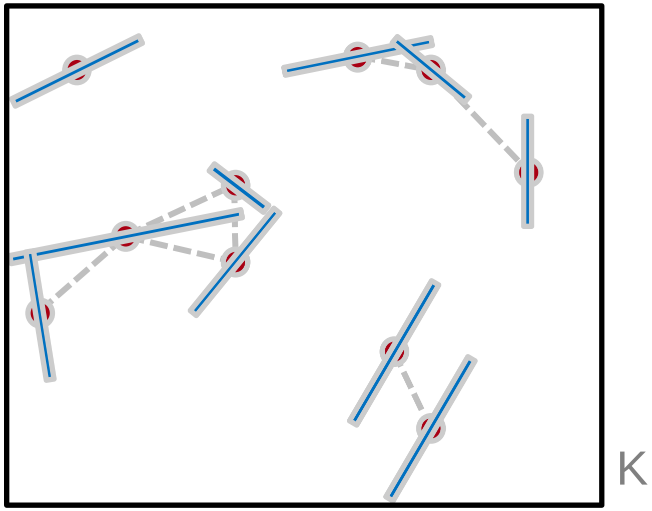


Background on point process

Random configurations of points distributed in a bounded domain K

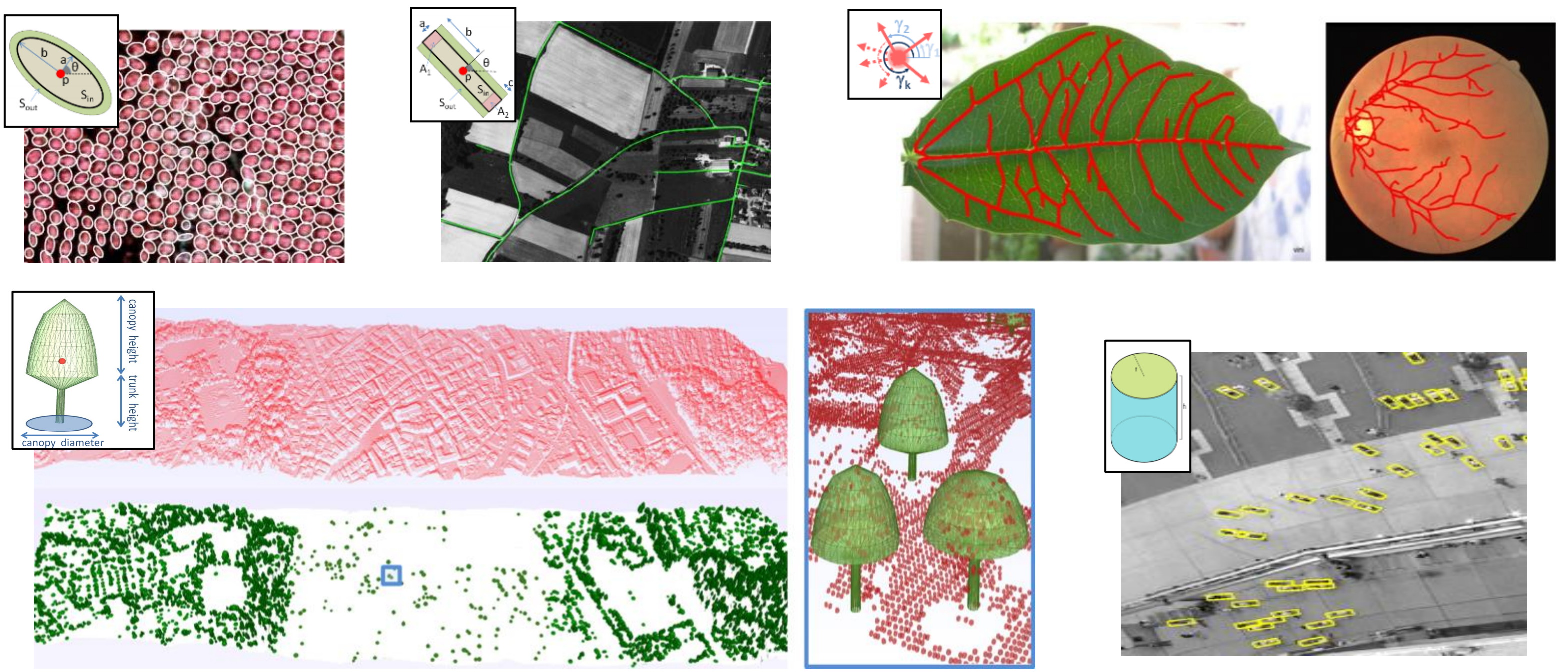
Interesting characteristics

- #points is a random variable
- can be guided by a density *h*
- with spatial interactions
- **each point can be associated with a parametric object**



Applications

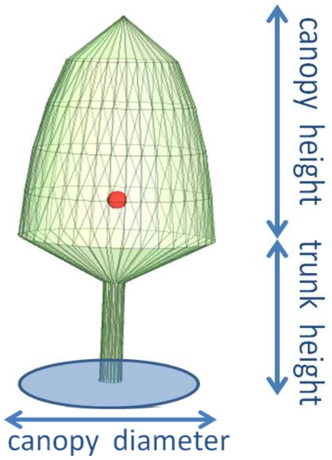
Parametric object detection



Three important ingredients

A parametric object

- points = object centroids
- some additional parameters



Three important ingredients

A parametric object

An energy U

- measures the quality of an object configuration
- specifies the density h of the process $h(.) \propto \exp -U(.)$

Three important ingredients

A parametric object

An energy U

- measures the quality of an object configuration
- specifies the density h of the process $h(.) \propto \exp -U(.)$
- typical form:

$$\forall \boldsymbol{x} \in \mathcal{C}, \quad U(\boldsymbol{x}) = \underbrace{\sum_{x_i \in \boldsymbol{x}} D(x_i)}_{\text{Data term}} + \underbrace{\sum_{x_i \sim x_j} V(x_i, x_j)}_{\text{Pairwise interactions}}$$

Three important ingredients

A parametric object

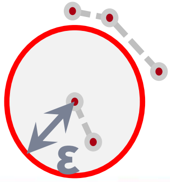
An energy U

- measures the quality of an object configuration
- specifies the density h of the process $h(.) \propto \exp -U(.)$
- typical form:

$$\forall \boldsymbol{x} \in \mathcal{C}, \quad U(\boldsymbol{x}) = \underbrace{\sum_{x_i \in \boldsymbol{x}} D(x_i)}_{\text{Data term}} + \underbrace{\sum_{x_i \sim x_j} V(x_i, x_j)}_{\text{Pairwise interactions}}$$

Markovian property: interactions restricted to a local neighborhood

$$x_i \sim x_j = \{(x_i, x_j) \in \boldsymbol{x}^2 : i > j, ||x_i - x_j||_2 < \epsilon\}$$



Three important ingredients

A parametric object

An energy U

A sampler

- Find an approximate solution of the global minimum of U typically RJMCMC sampler [Green95]

Three important ingredients

A parametric object

An energy U

A sampler

- Find an approximate solution of the global minimum of U typically RJMCMC sampler [Green95]
- **Principle:** iterative mechanism that simulates a discrete Markov chain on the configuration space

Three important ingredients

A parametric object

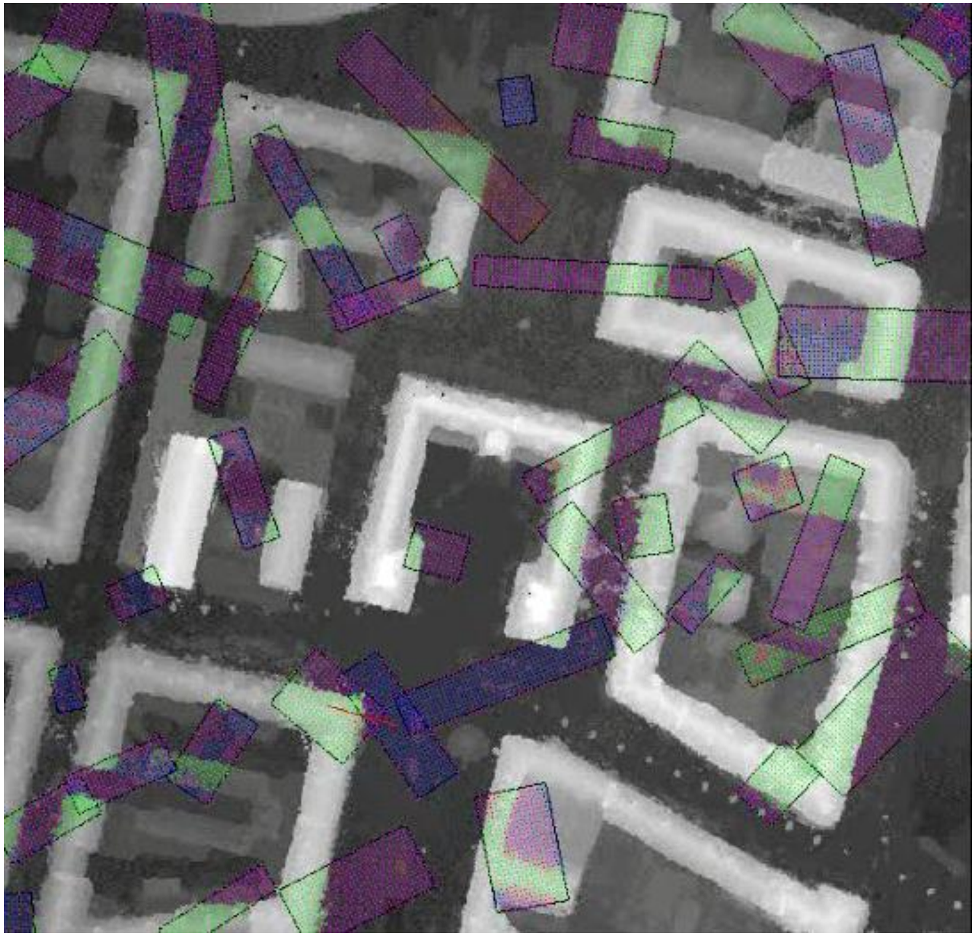
An energy U

A sampler

- Find an approximate solution of the global minimum of U typically RJMCMC sampler [Green95]
- **Principle:** iterative mechanism that simulates a discrete Markov chain on the configuration space
- **At each iteration,**
  (i) proposition of a local modification
  (ii) acceptation/rejection of the modification depending on energy variation, proposal densities and a relaxation parameter

Three important ingredients

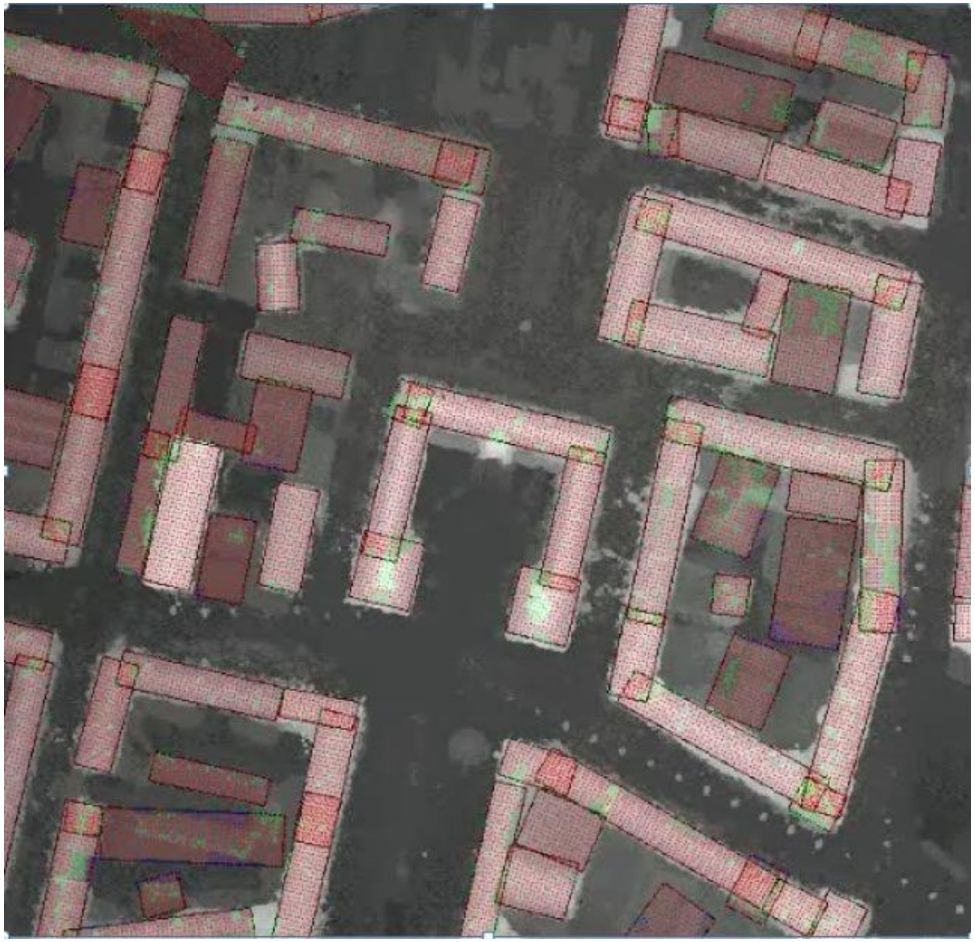
An example: extracting buildings from Elevation maps



(time ×150)

Three important ingredients

An example: extracting buildings from Elevation maps



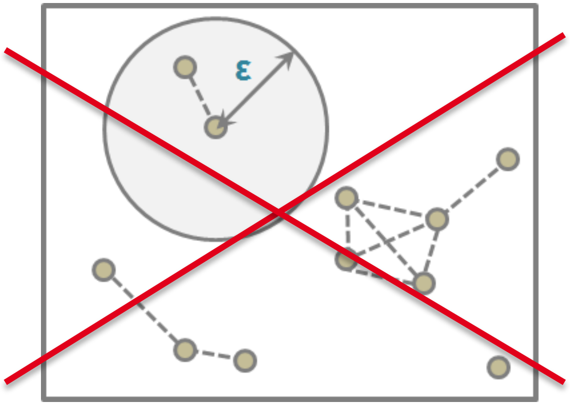
Geometric structures not guaranteed by construction!

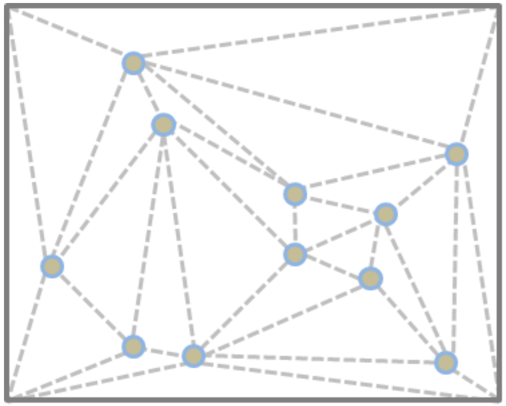
(time ×150)

Delaunay neighborhood

Idea: use Delaunay triangulation to define neighboring relationship (instead a traditional Euclidean distance)

$$p_i \sim_D p_j = \{(p_i, p_j) \in \boldsymbol{p}^2 : (p_i, p_j) \in C_2(\boldsymbol{p})\}$$





Why is it interesting?

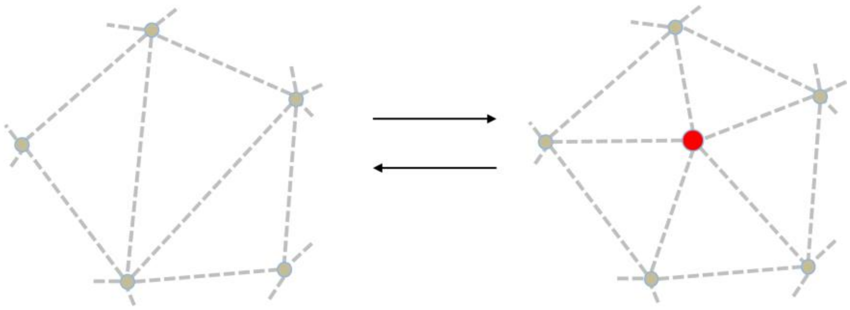
Each configuration relies on space decomposition that can be used as a mean to sample points but also as goal to segment data

- Parameter-free neighborhood

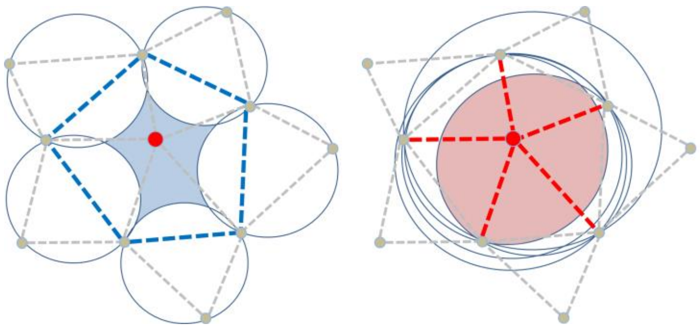
Why is it interesting?

Each configuration relies on space decomposition that can be used as a mean to sample points but also as goal to segment data

- Parameter-free neighborhood
- Efficient sampling



add/delete a vertex

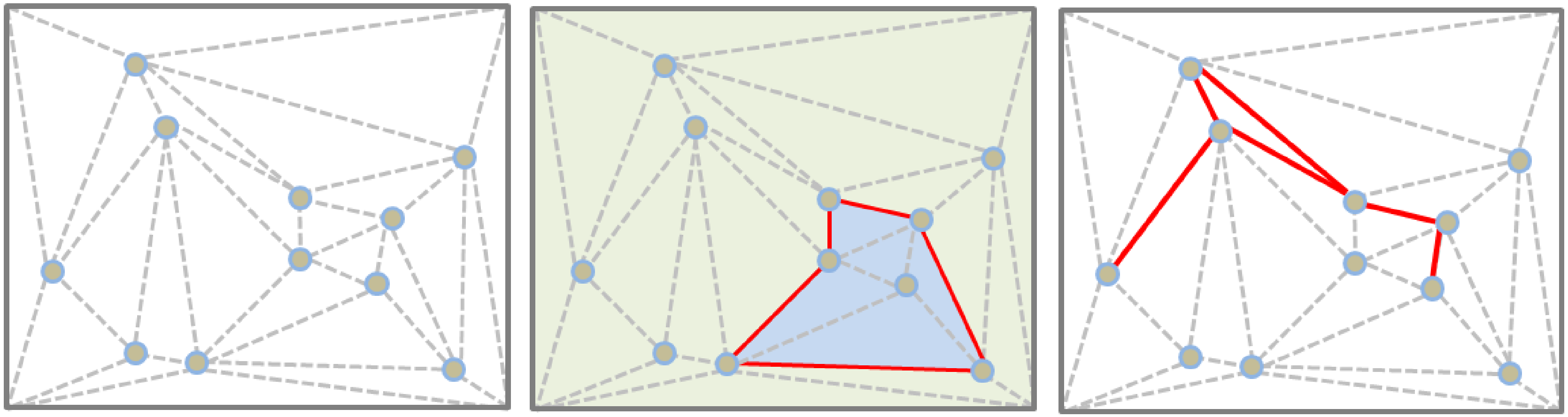


relocate a vertex

Why is it interesting?

Each configuration relies on space decomposition that can be used as a mean to sample points but also as goal to segment data

- Parameter-free neighborhood
- Efficient sampling
- flexibility for a large range of applications

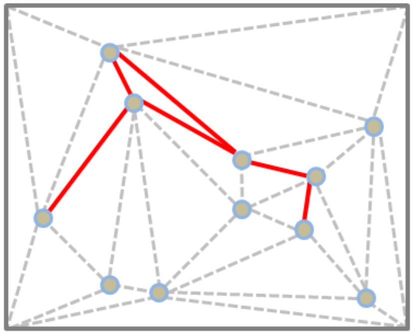


Application to line-network extraction

Energy formulation

A configuration $x = (p, m)$ a set of points and some additional parameter on points, edges or facets of the triangulation

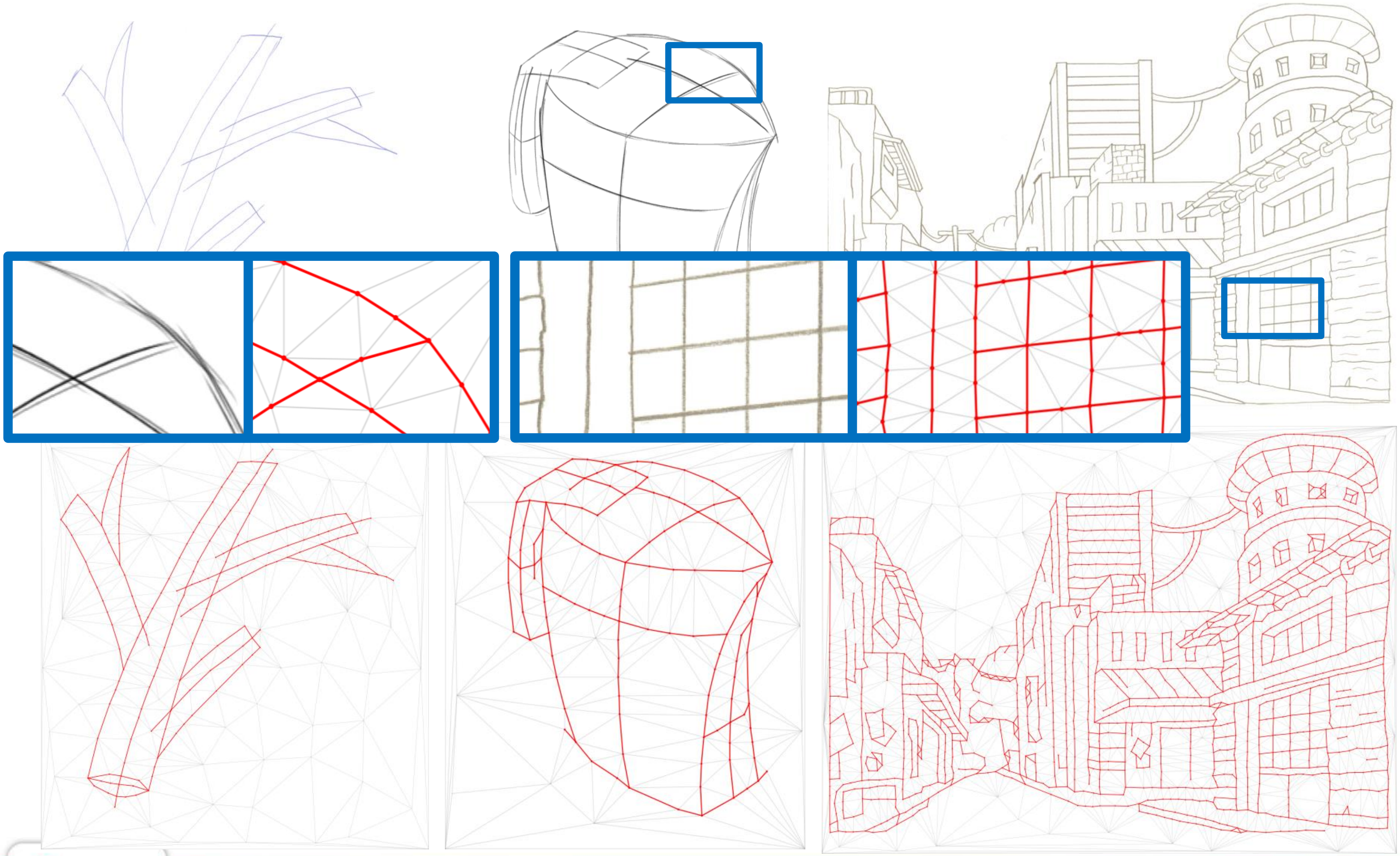
Energy of the form $U(x) = U_{fidelity}(x) + U_{prior}(x)$



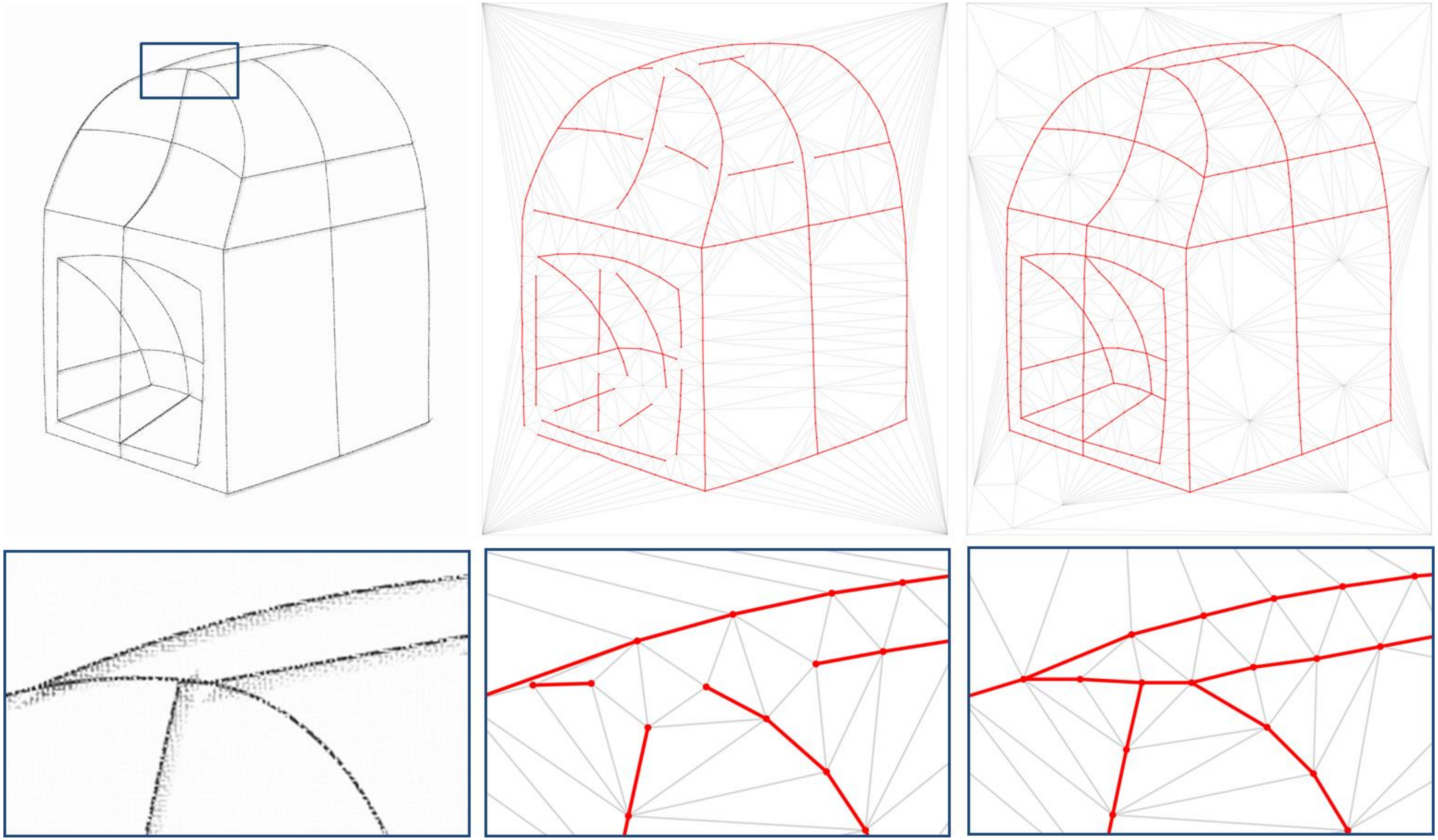
Example with line-network extraction

- $m = (m_e)_{e \in C_2(p)}$ with $m_e \in \{0, 1\}$
- $U_{fidelity}$: coherence with data (active edges should align with strong gradients)
- U_{prior} : penalty for short edges + penalty for badly connected edges
- Sampling: RJMCMC with points distributed with density following an image gradient

Application to line-network extraction

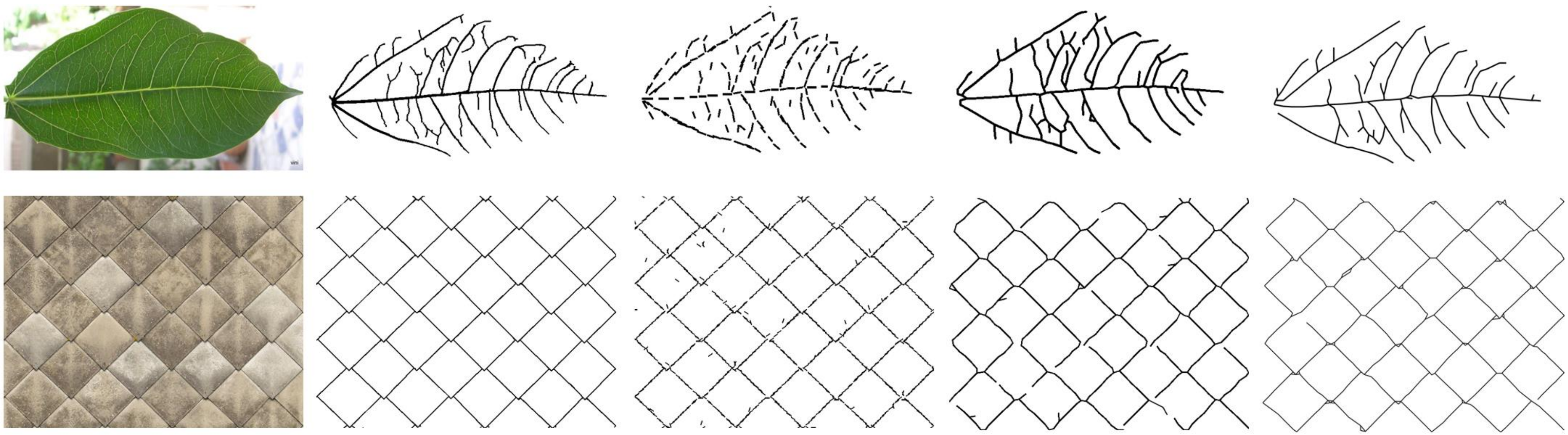


Application to line-network extraction



input sequentially jointly

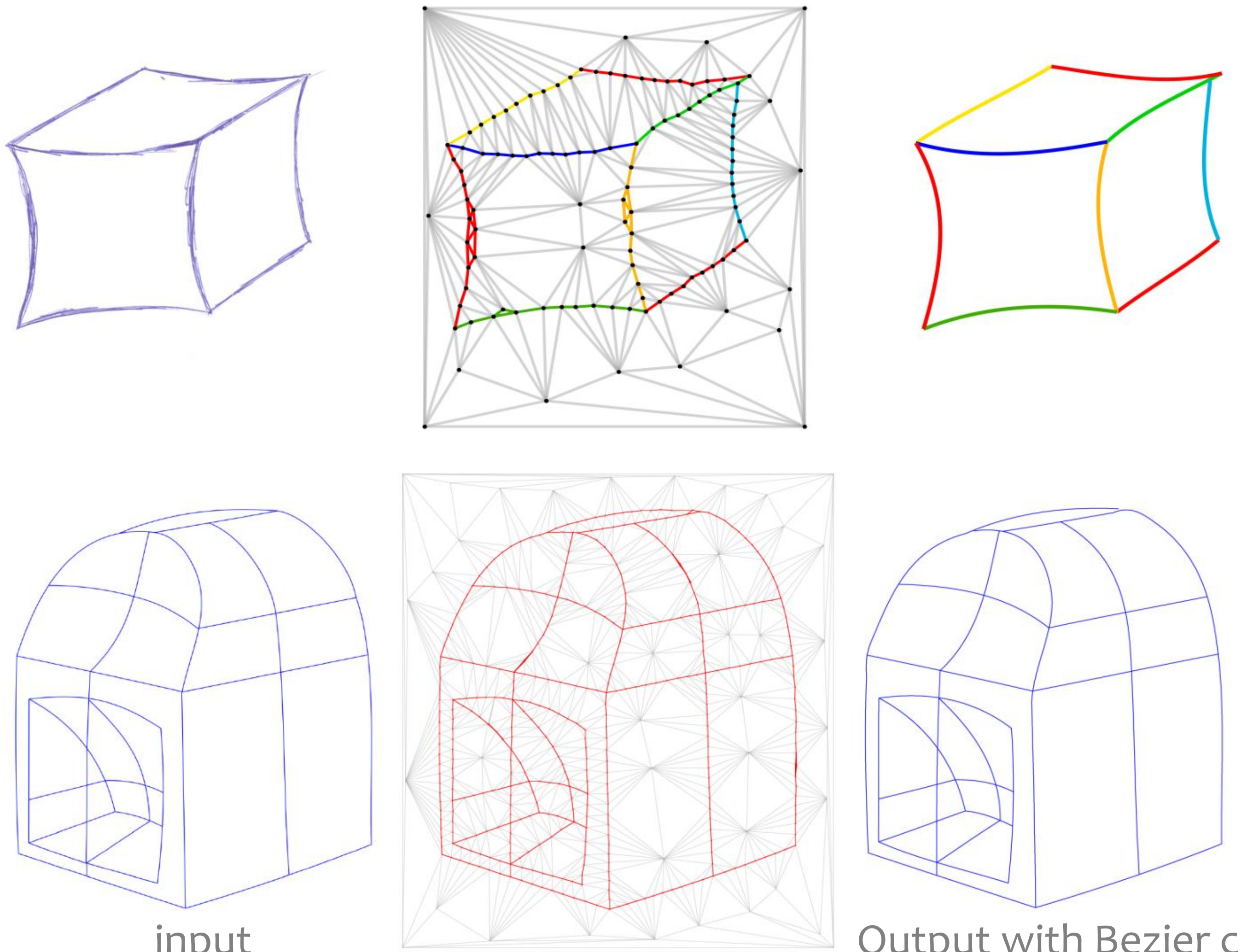
Application to line-network extraction



Input | GT | Marked point process [Verdie et al, IJCV14] | Junction point process [Chai et al, CVPR13] | Ours

		Precision	F-measure	Time
Leaf	Junction-point process	0.59	0.64	73s
	Marked point process	0.76	0.70	33s
	ours	**0.79**	**0.73**	20s
Tiles	Junction-point process	0.46	0.54	227s
	Marked point process	0.67	0.72	103s
	ours	**0.70**	**0.74**	70s

Application to line-network extraction



input

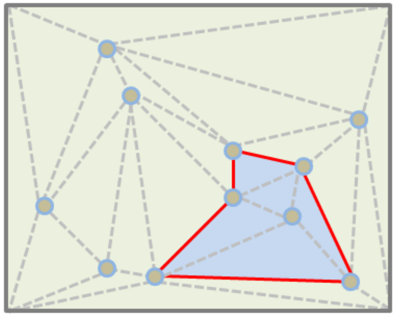
Output with Bezier curves

Application to object contouring

Application to object contouring

A configuration $x = (p, m)$ a set of points and some additional parameter on points, edges or facets of the triangulation

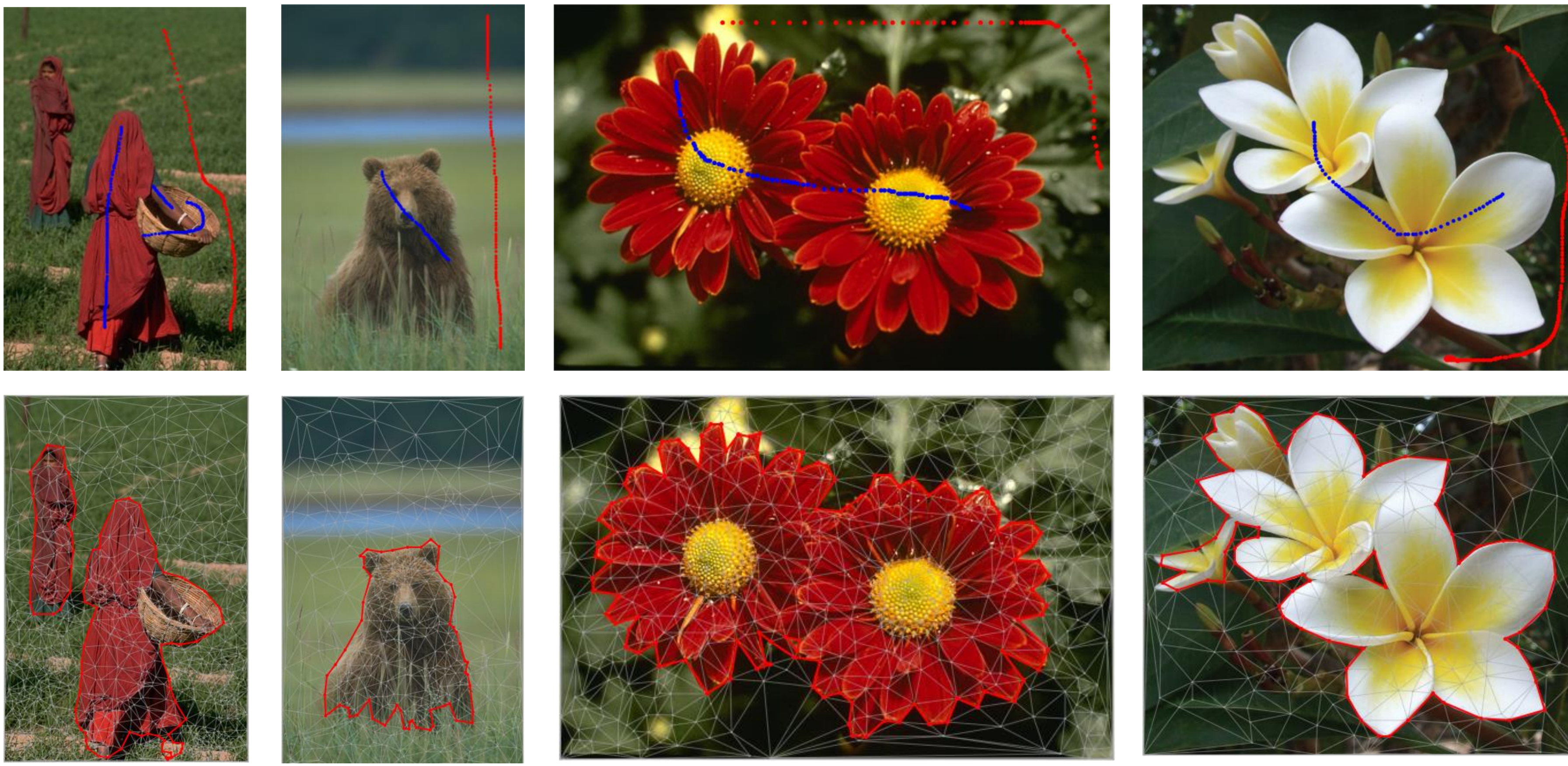
Energy of the form $U(x) = U_{fidelity}(x) + U_{prior}(x)$



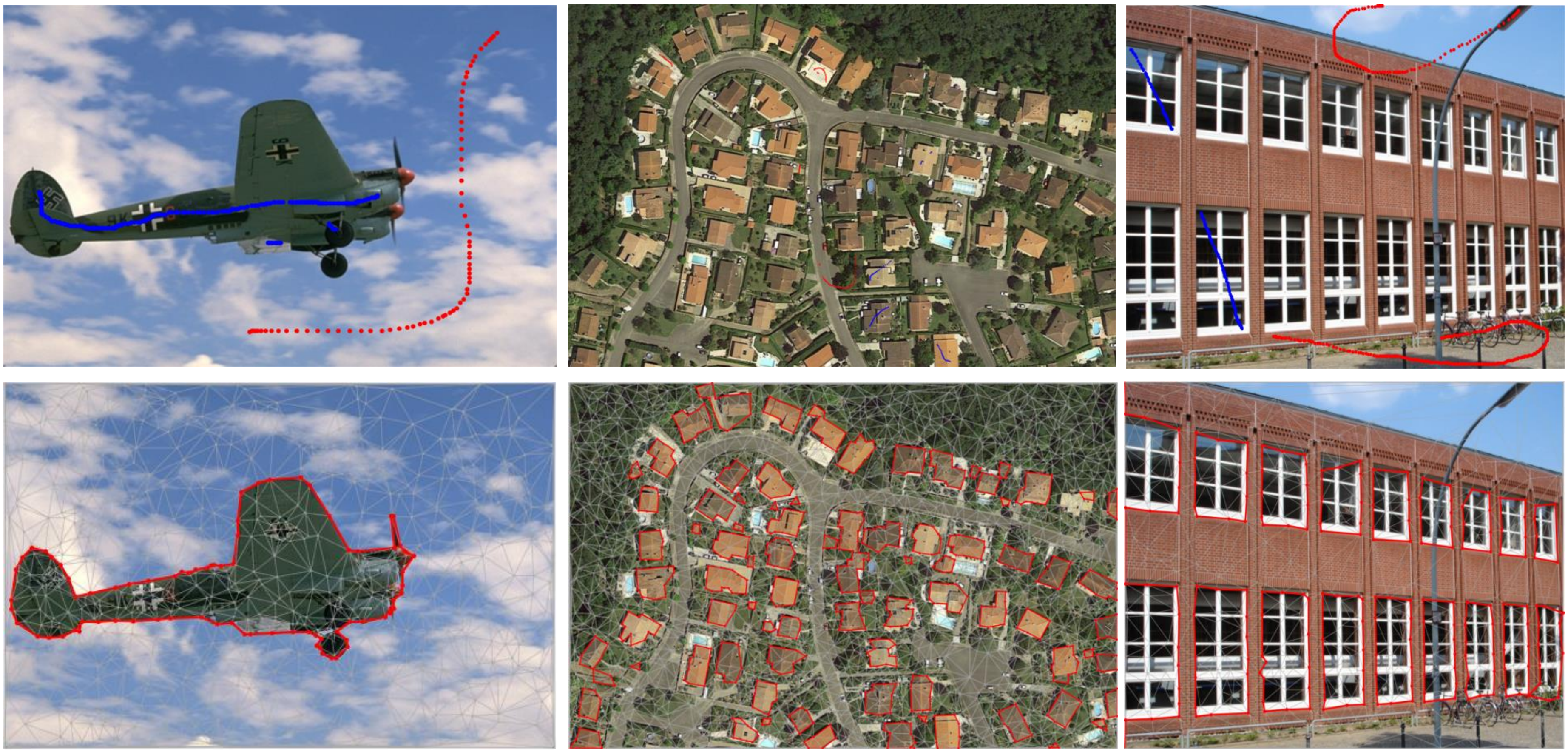
Example with object countouring

- $m = (l_f)_{f \in C_3(p)}$ with $l_f = 0, 1$
- $U_{fidelity}$: radiometric coherence inside each facet
- U_{prior} : penalty for short edges + label smoothness for adjacent facets
- Sampling: RJMCMC with points distributed with density following an image gradient

Application to object contouring

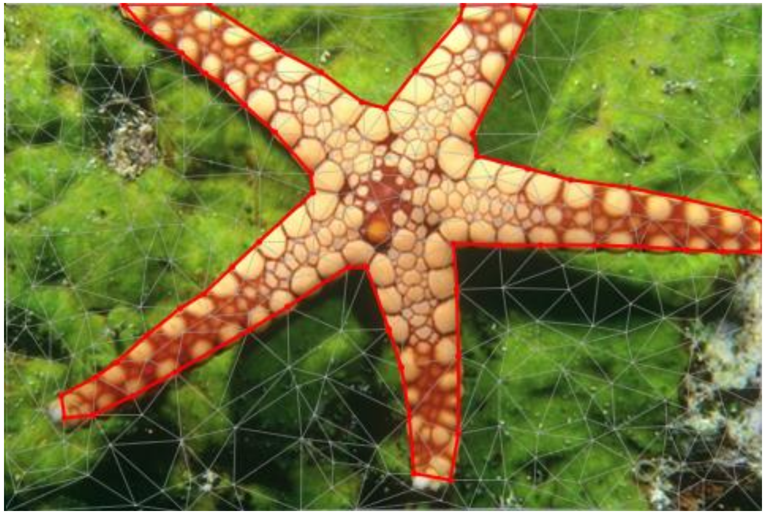


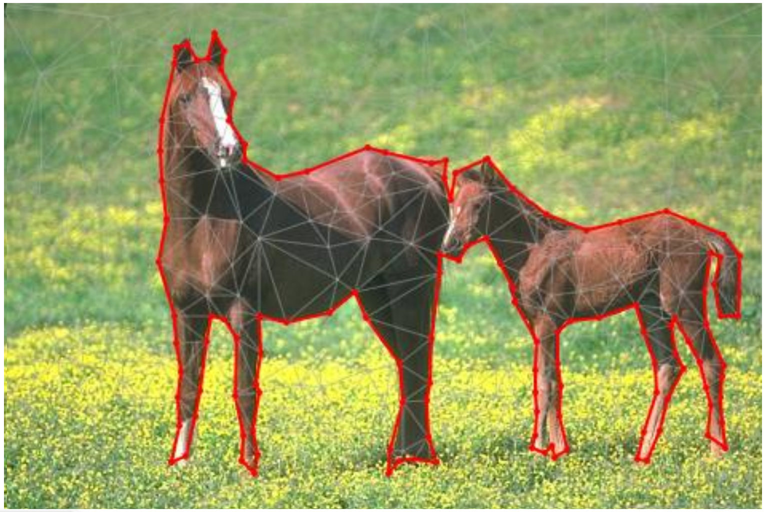
Application to object contouring



Application to object contouring

Applications to object contouring





Application to object contouring

low point density

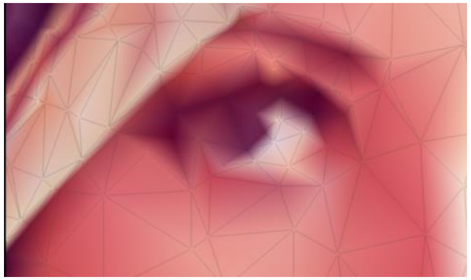
high point density

Application to image compression

Application to image compression

A configuration $x = (p, m)$ a set of points and some additional parameter on points, edges or facets of the triangulation

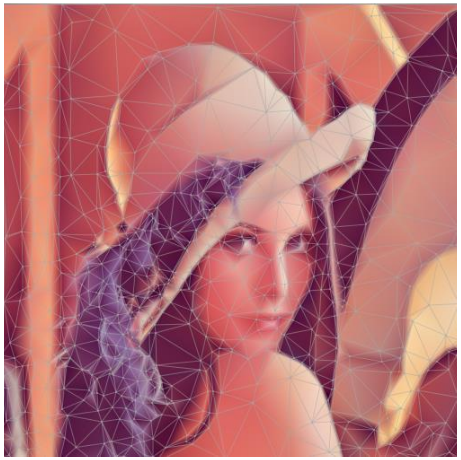
Energy of the form $U(x) = U_{fidelity}(x) + U_{prior}(x)$

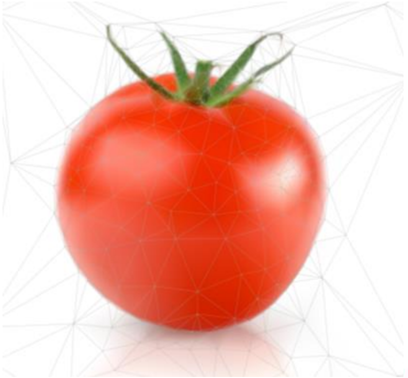


Example with image compression

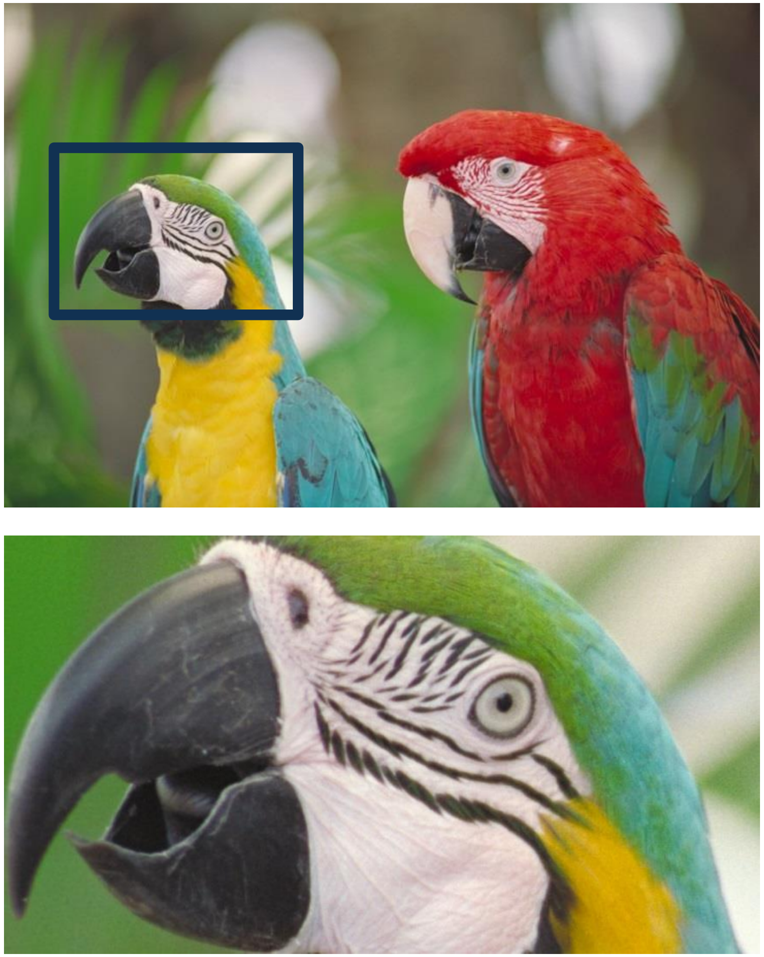
- $m = (c_p)_{p \in \boldsymbol{p}}$ where c_p is a RGB color
- $U_{fidelity}$: per-pixel error between input/output
- U_{prior} : penalty for high number of points
- Sampling: RJMCMC with points distributed with density following an image gradient

Application to image compression

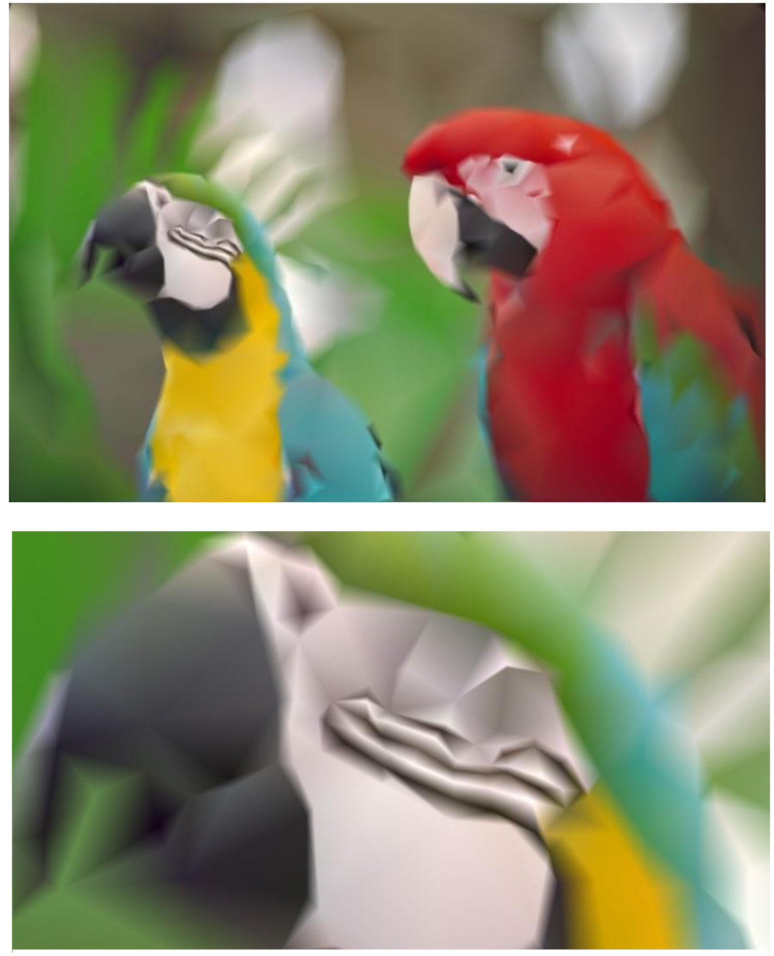




Application to image compression



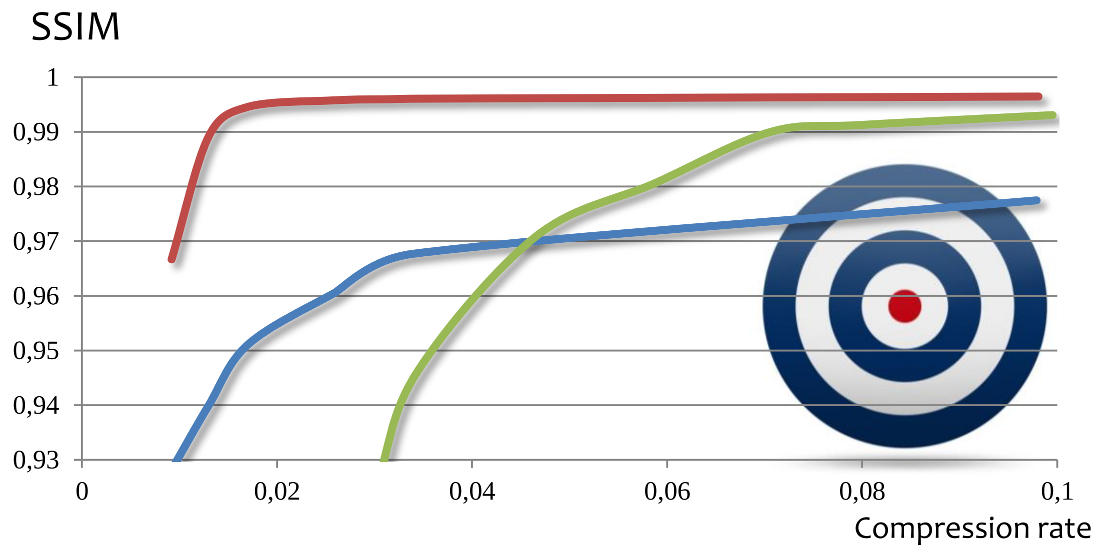
Input (6,3Mpix)

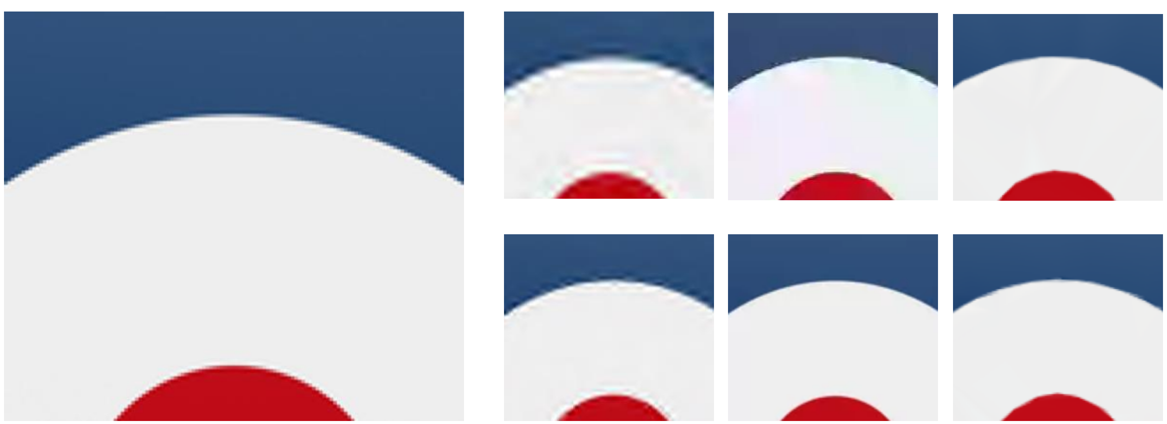


Output (700 vertices)

Application to image compression

on synthetic image





3% compression

10% compression

original

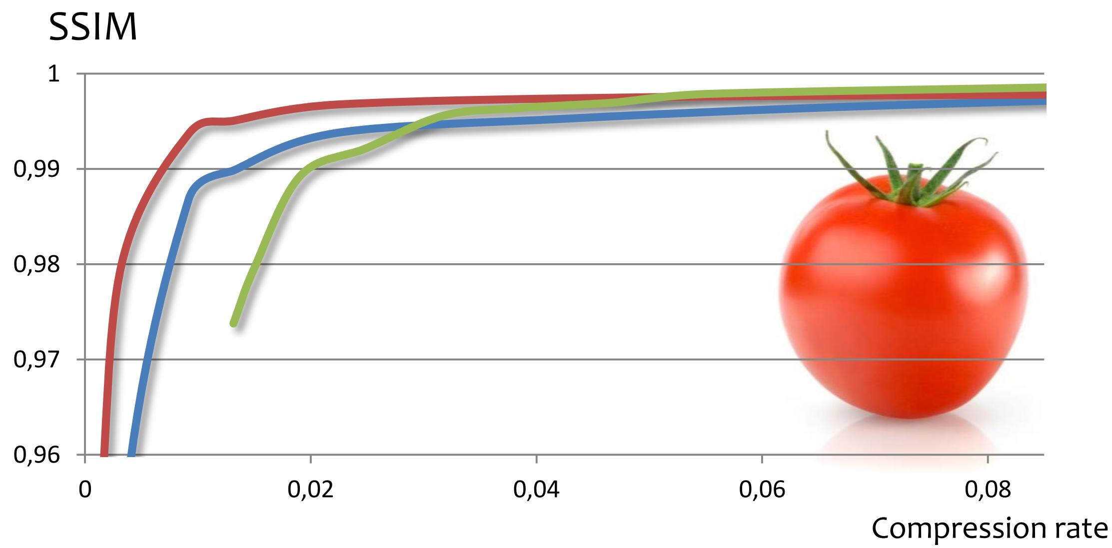
JPEG2000

WEBP

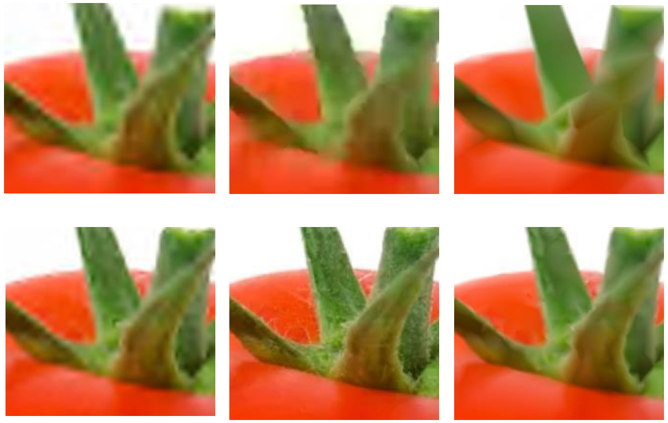
Our

Application to image compression

on studio picture



original



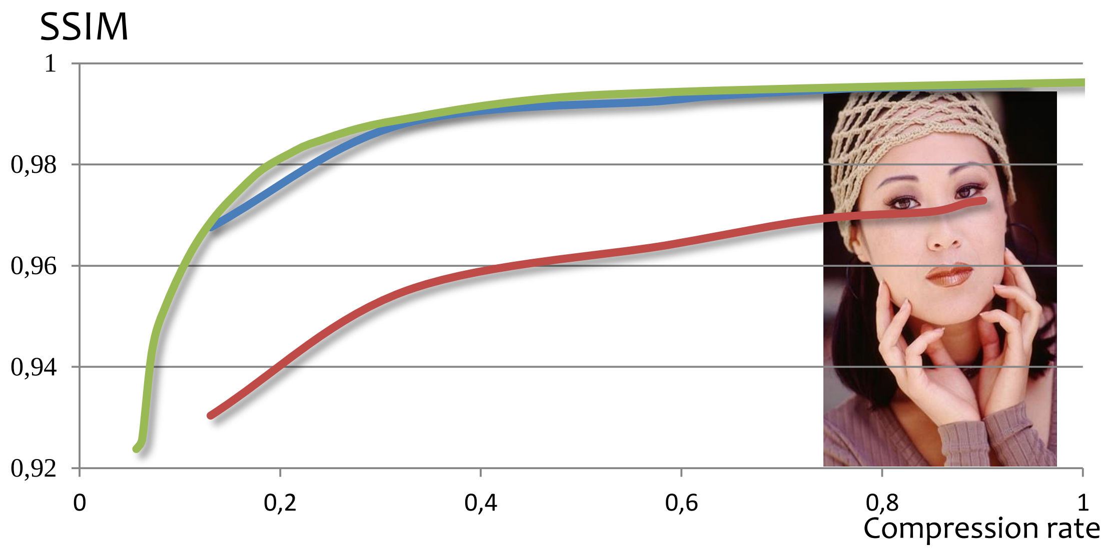
0.1% compression

5% compression

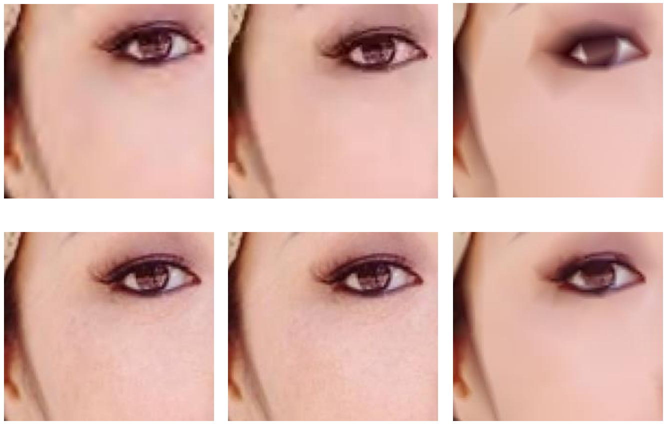
JPEG2000 WEBP Our

Application to image compression

on studio picture



original



13% compression

70% compression

JPEG2000 WEBP Our

Conclusion

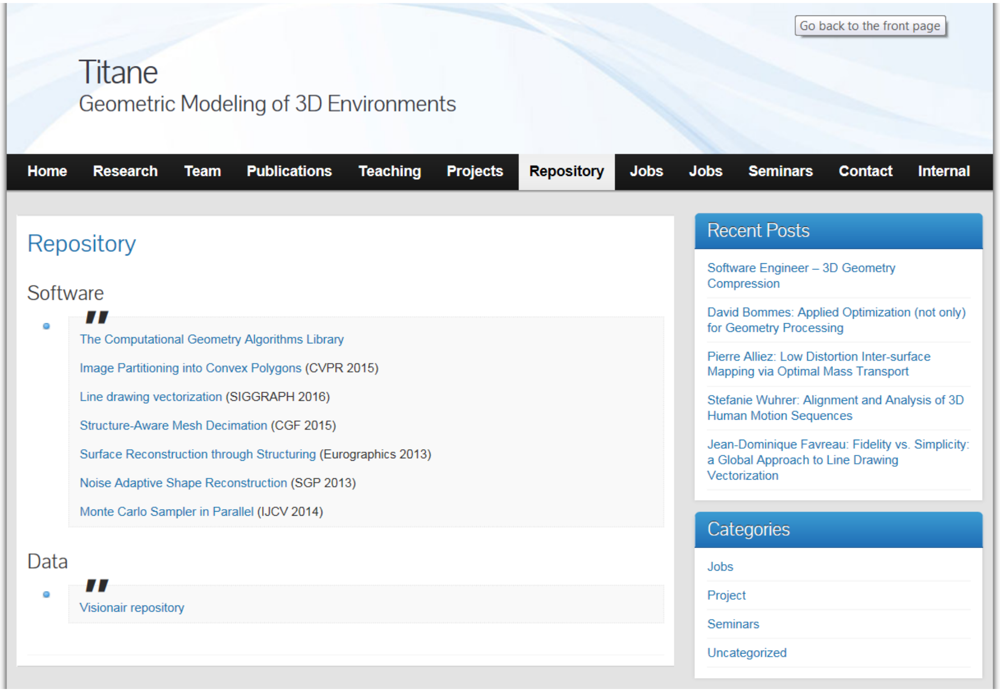
Partitioning images into geometric data structures:

- fast preprocessing
- man-made objects and scenes
- Scalability
- easy-to-use
- comes with geometric guarantees (cell adjacency, convexity..)

Extensions

- Integrate image partitioning into the application process
- more types of shape
- 3D

Code online



https://team.inria.fr/titane/software/