3D Reconstruction of objects and scenes

Florent Lafarge
Inria Sophia Antipolis - Mediterranee

Contents

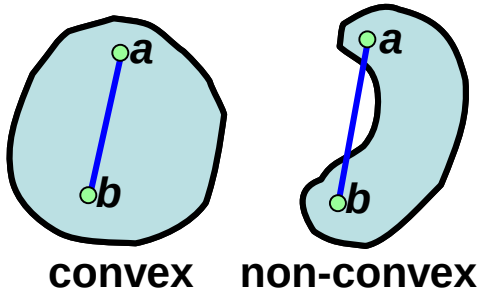
- Reconstruction of freeform objects
- Reconstruction of piecewise-planar objects
- City modeling

Reconstruction of freeform objects

- Explicit methods
- Implicit methods

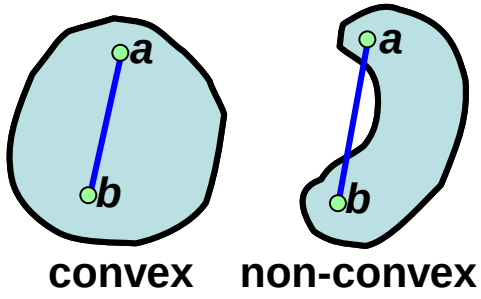
Convex Hulls:

A set S is *convex* if for any two points $a,b \in S$, the line segment between a and b is also in S.

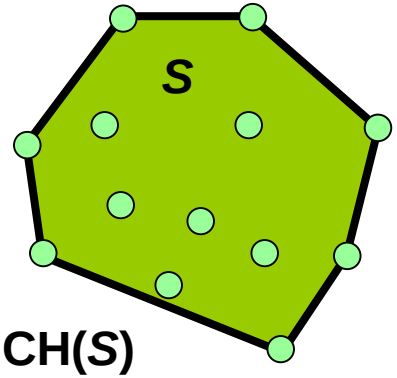


Convex Hulls:

A set S is *convex* if for any two points $a,b \in S$, the line segment between a and b is also in S.

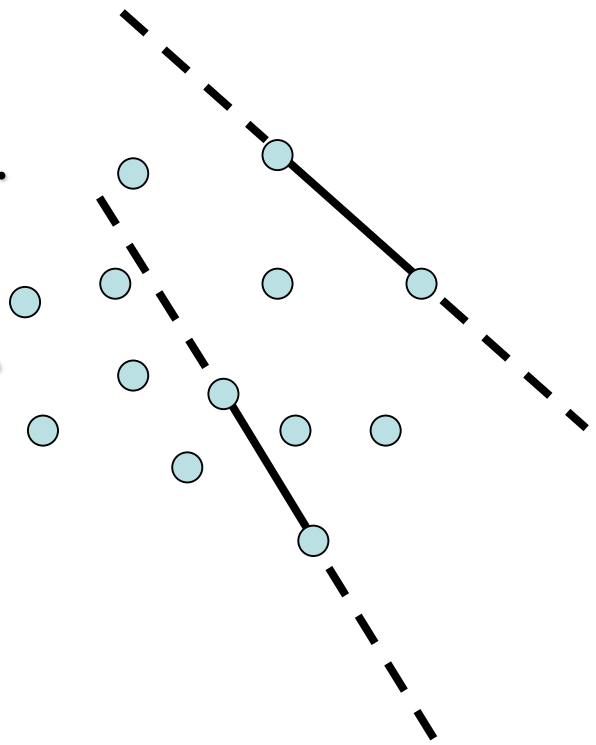


The *convex hull* of a set of points is the smallest convex set containing S.



Convex Hulls: Naive construction

- Description:
  - For each pair of points construct its connecting segment and *supporting line*.
  - Find all the segments whose supporting lines divide the plane into two halves, such that one half plane contains *all* the other points.
  - Construct the convex hull out of these segments.



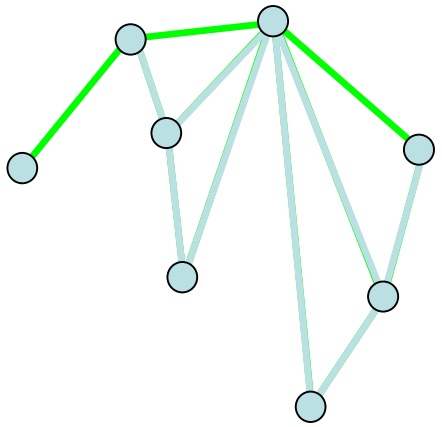
- Time complexity:
  - All pairs:

$$O(\binom{n}{2}) = O(\frac{n(n-1)}{2}) = O(n^2)$$

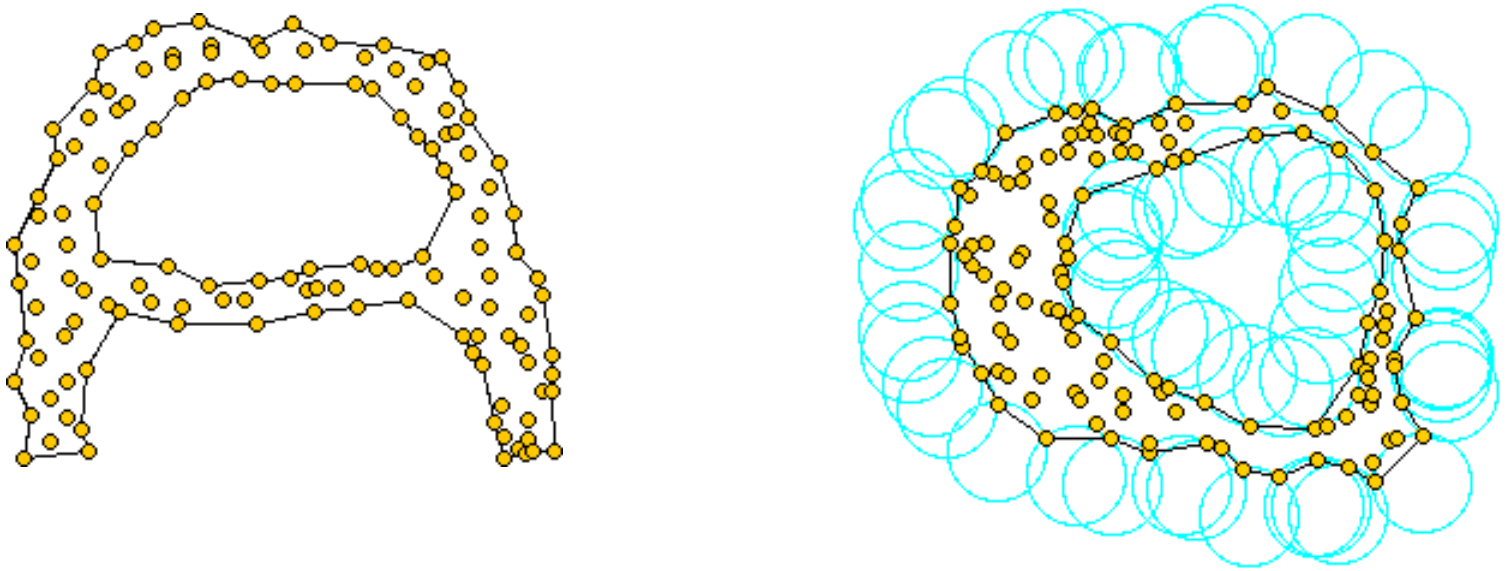
  - Check all points for each pair: O(n)
  - Total: O(n^3)

Convex Hulls: Graham's scan

- Algorithm:
  - Sort the points according to their *x* coordinates.
  - Construct the upper boundary by scanning the points in the sorted order and performing only "right turns".
  - Construct the lower boundary (with "left turns").
  - Concatenate the two boundaries.
- Time Complexity: O(*n*log*n*)
- May be implemented using a stack

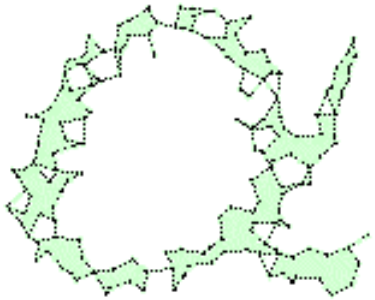


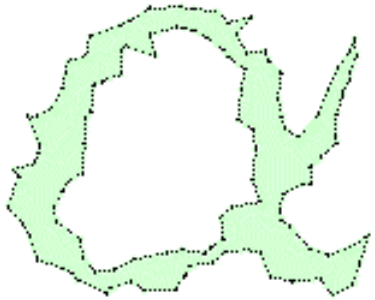
Alpha shapes:



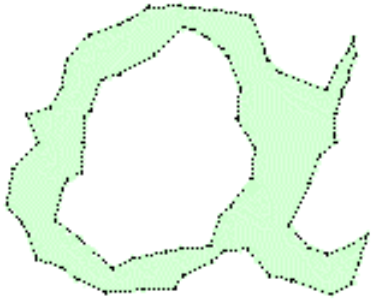
The space generated by point pairs that can be touched by an empty disc of radius alpha.

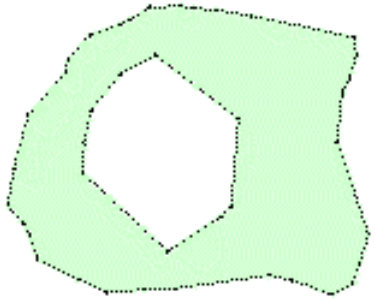
Alpha shapes:

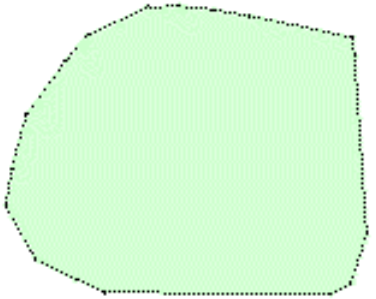




$\alpha = 0$ Alpha Controls the desired level of detail.







$\alpha = \infty$

Delaunay Triangulation:

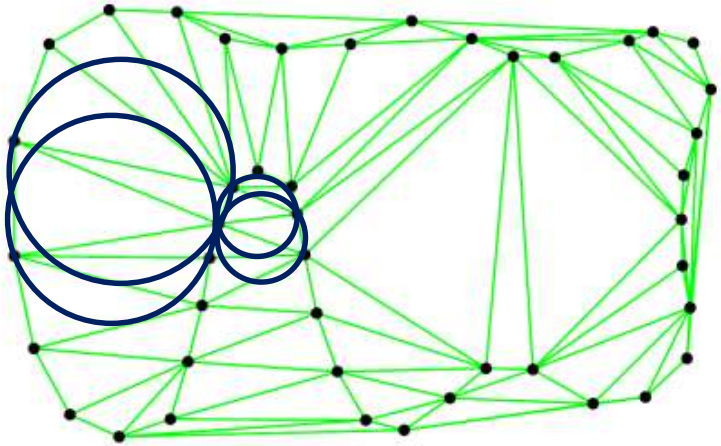
A *Delaunay Triangulation* of *S* is the set of all triangles with vertices in *S* whose circumscribing circle contains no other points in *S**.

Compactness Property:

This is a triangulation that maximizes the min angle of all

The angles of the triangles

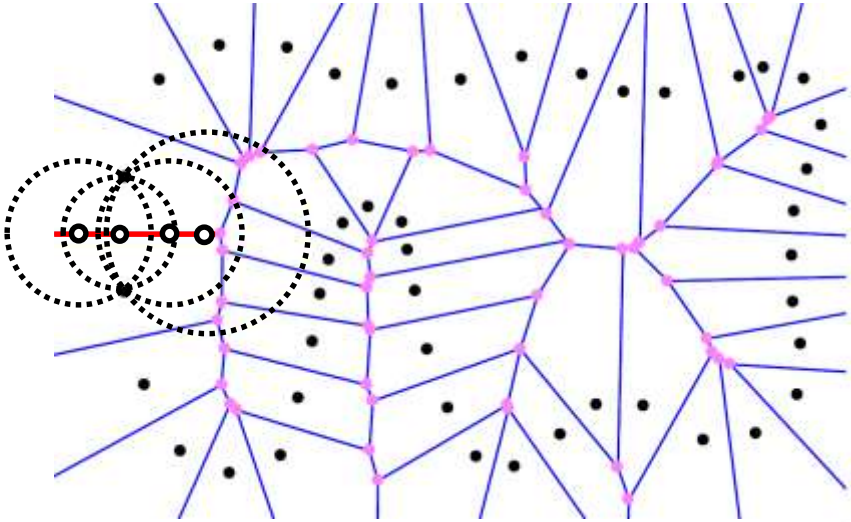
(avoid skinny triangles)



Voronoi Diagrams:

The *Voronoi Diagram* of S is a partition of space into regions $V(p)$ ($p \in S$) such that all points in $V(p)$ are closer to p than any other point in S.

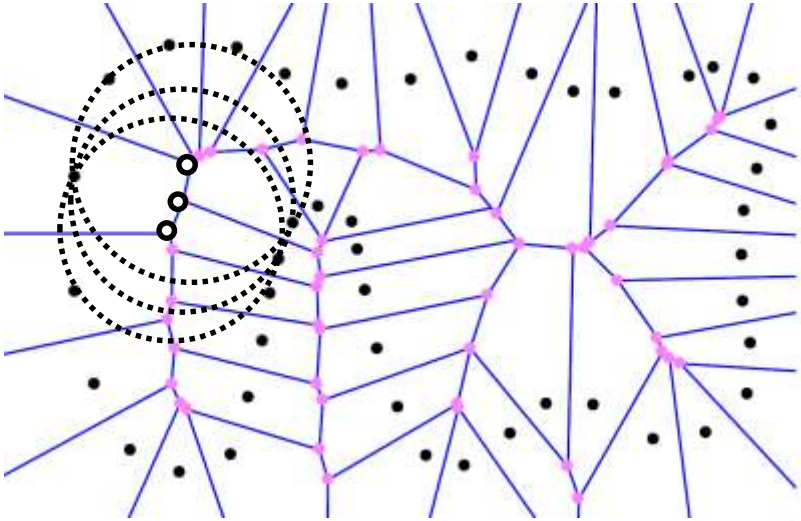
For a point on an edge, we can draw an empty circle that only touches the points in S separated by the edge.



Voronoi Diagrams:

The *Voronoi Diagram* of S is a partition of space into regions $V(p)$ ($p \in S$) such that all points in $V(p)$ are closer to p than any other point in S.

For a vertex, we can draw an empty circle that just touches the three points in S around the vertex.

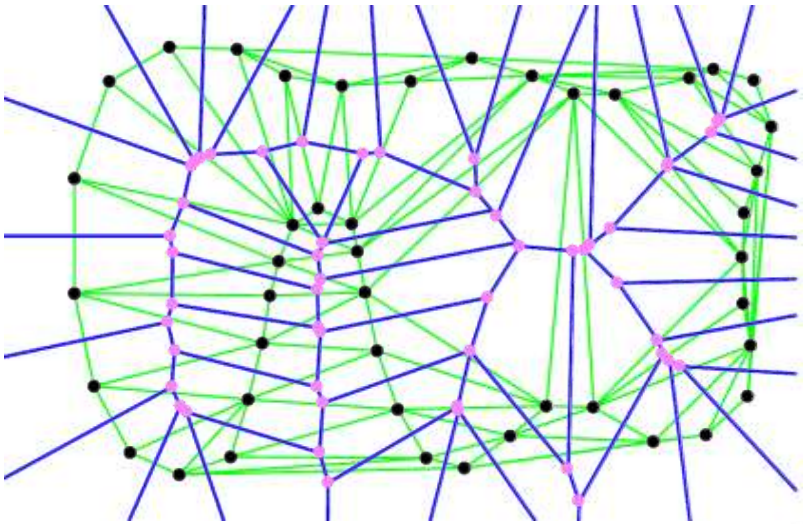


Voronoi Diagrams:

The *Voronoi Diagram* of S is a partition of space into regions $V(p)$ ($p \in S$) such that all points in $V(p)$ are closer to p than any other point in S.

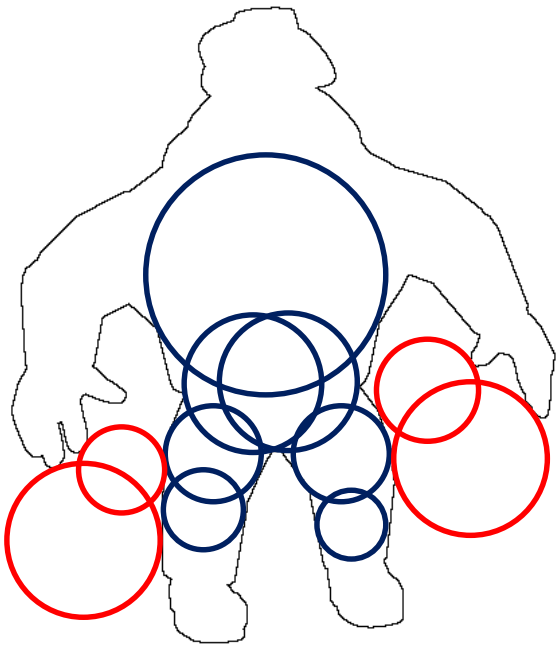
Duality:

Each Voronoi vertex is in one-to-one correspondence with a Delaunay triangle.



Medial Axis:

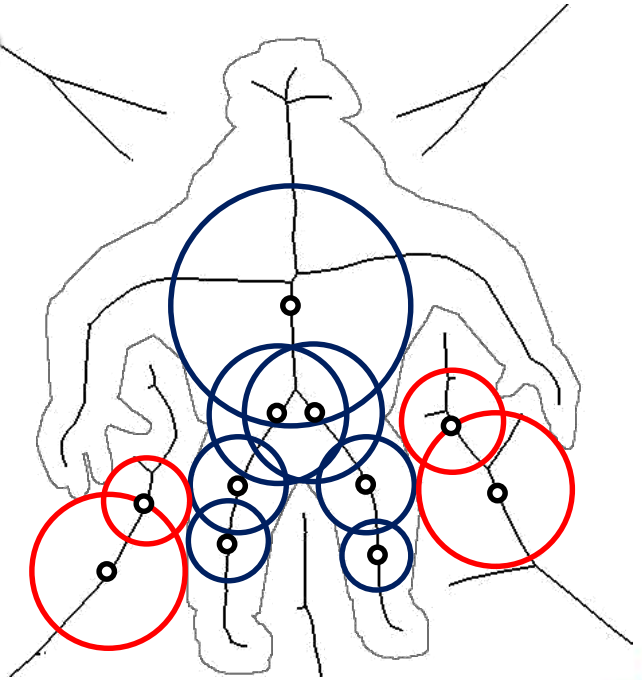
For a shape (curve/surface) a *Medial Ball* is a circle/sphere that only meets the shape tangentially, in at least two points.



Medial Axis:

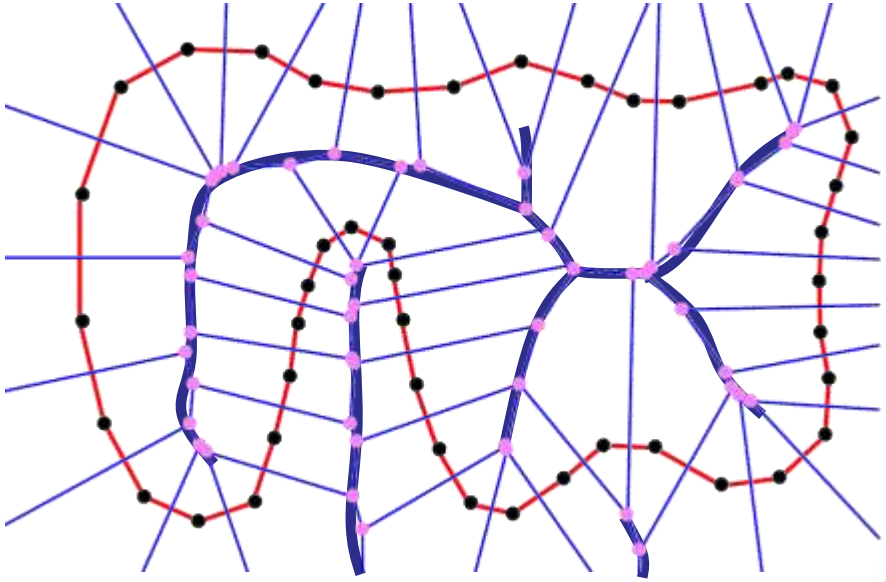
For a shape (curve/surface) a *Medial Ball* is a circle/sphere that only meets the shape tangentially, in at least two points.

The centers of all such balls make up the *medial axis/skeleton*.



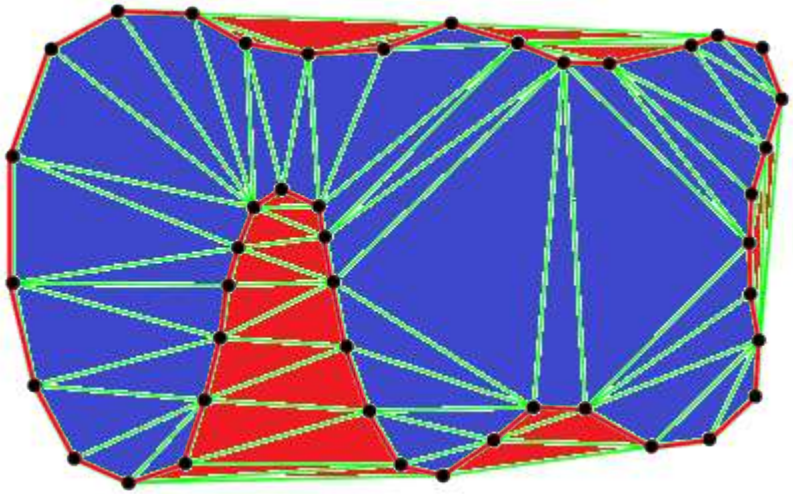
Observation*:

For a reasonable point sample, the medial axis is well-sampled by the Voronoi vertices.



Given a set of points, we can construct the Delaunay triangulation.

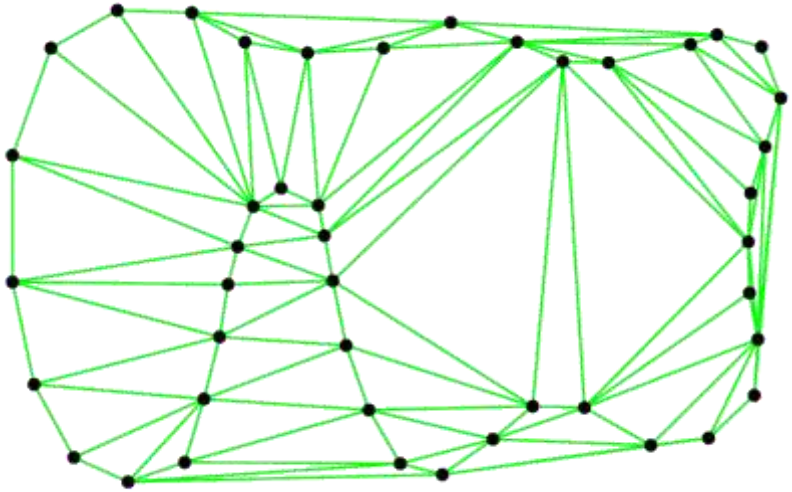
If we label each triangle as inside/outside, then the surface of interest is the set of edges that lie between inside and outside triangles.



Q: How to assign labels?

A: Spectral Partitioning

- Assign a weight to each edge indicating if the two triangles are likely to have the same label.



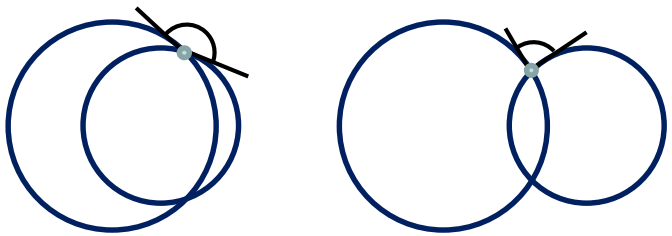
[Kolluri et al., 2004]

Assigning Edge Weights:

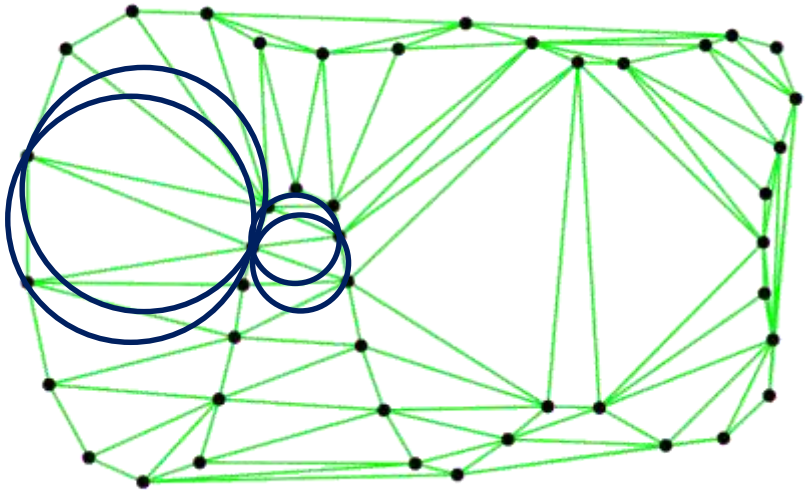
Q: When are triangles on opposite sides of an edge likely to have the same label?

A: If the triangles are on the same side, their circumscribing circles intersect deeply.

Use the angle of intersection to set the weight.



Large Weight Small Weight



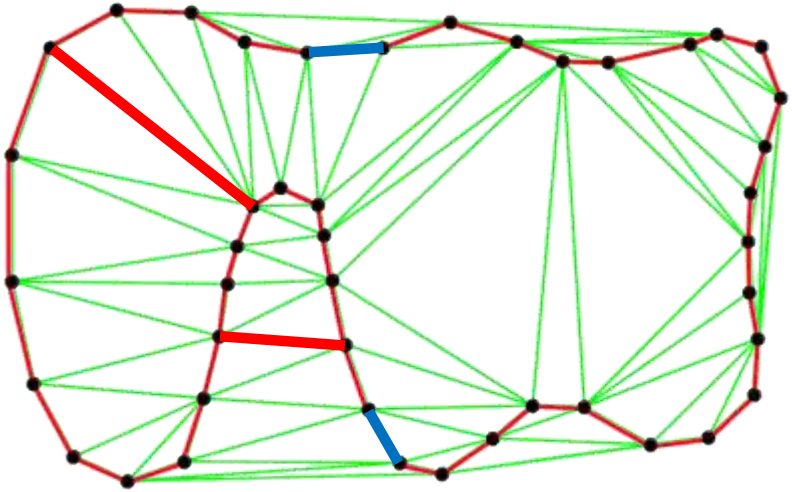
If we consider the Delaunay Triangulation of a point set, the shape boundary can be described as a subset of the Delaunay edges.

Q: How do we determine which edges to keep?

A: Two types of edges:

1. Those connecting adjacent points on the boundary
2. Those traversing the shape.

Discard those that traverse.



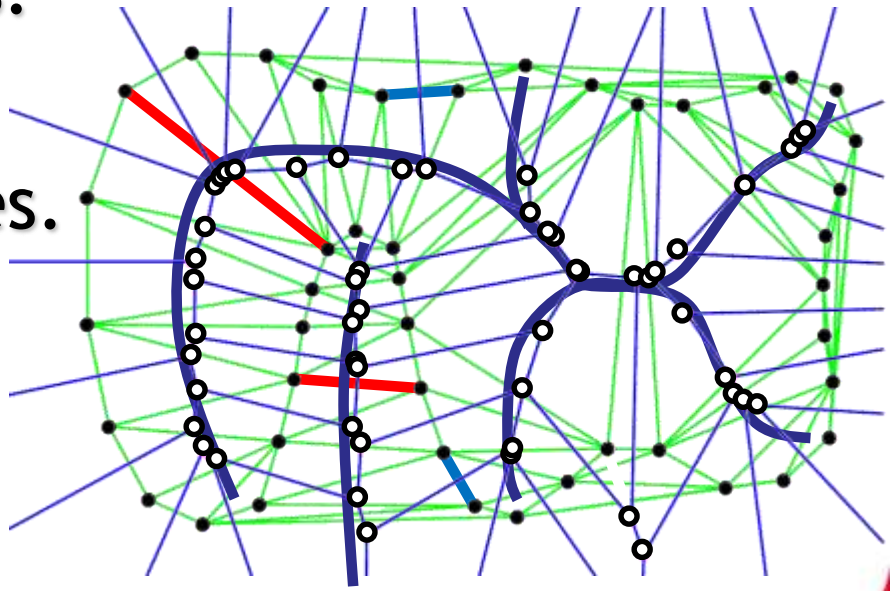
Observation:

Edges that traverse cross the medial axis.

Although we don't know the axis, we can sample it with the Voronoi vertices.

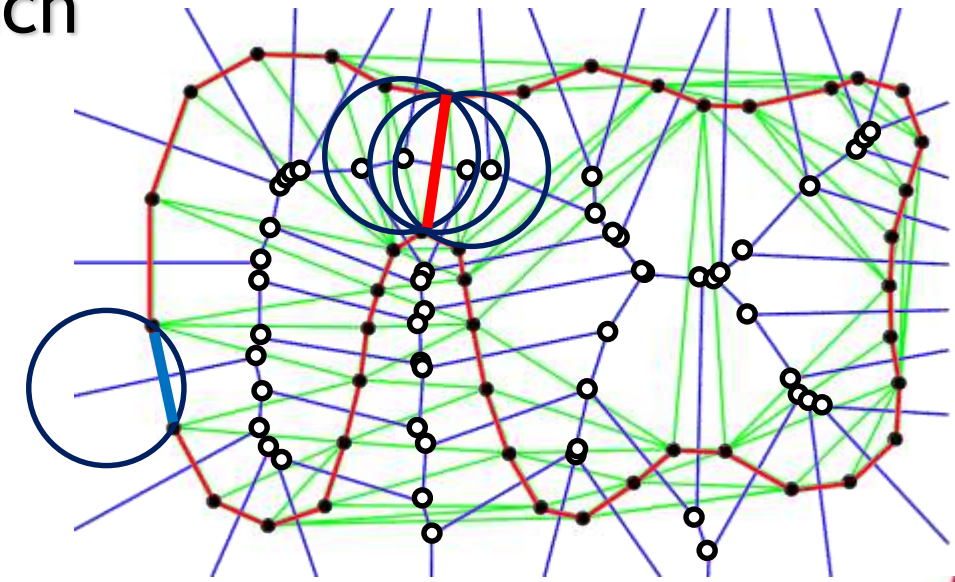
Edges that traverse must

be near the Voronoi vertices.



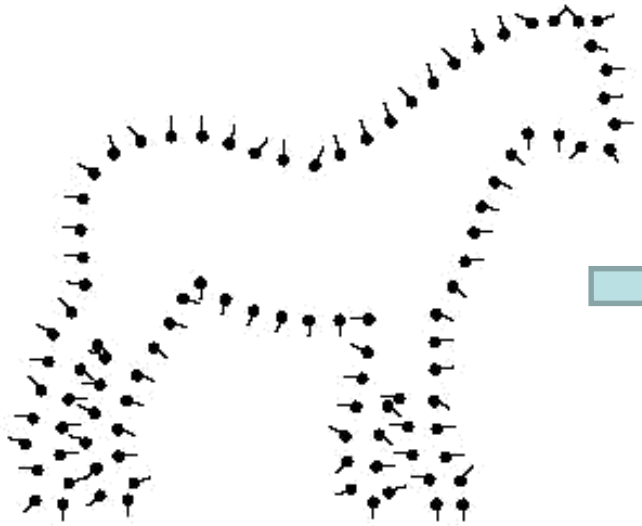
Algorithm:

1. Compute the Delaunay triangulation.
2. Compute the Voronoi vertices
3. Keep all edges for which there is a circle that contains the edge but no Voronoi vertices.

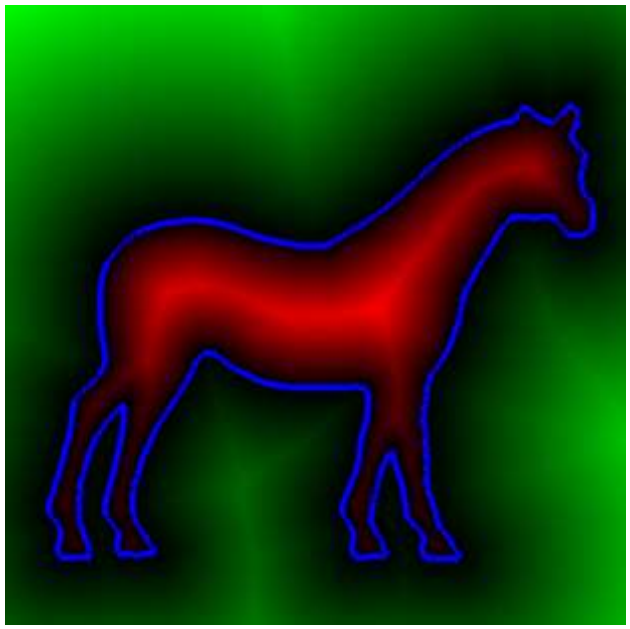


Key Idea:

- Use the point samples to define a function whose values at the sample positions are zero.



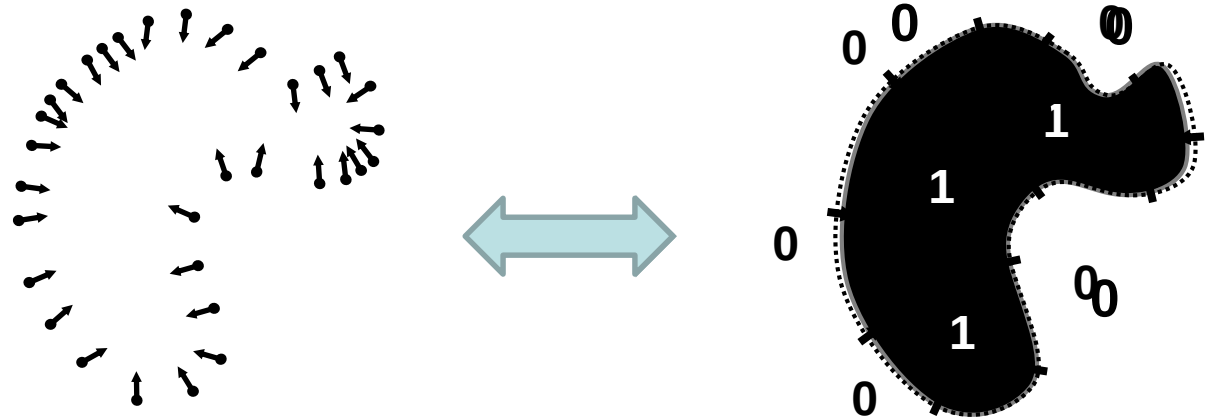
Sample Points



Reconstructs the indicator function of the surface and then extracts the boundary.

Q: How to fit the function to the samples?

A: Normals sample the function's gradients.

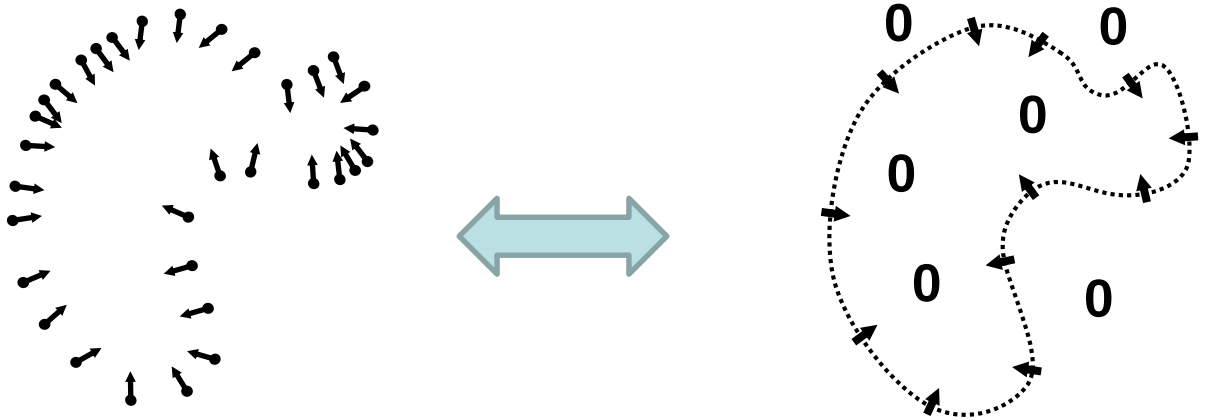


[Kazhdan et al., 2006]

To fit a scalar field F to the gradients $\vec{V}$:

$$\min_F \left\| \nabla F - \vec{V} \right\|^2$$

we take the divergence of both sides, which results in a Poisson equation: $\nabla \cdot (\nabla F) - \nabla \cdot \vec{V} = 0 \quad \Leftrightarrow \quad \Delta F = \nabla \cdot \vec{V}$

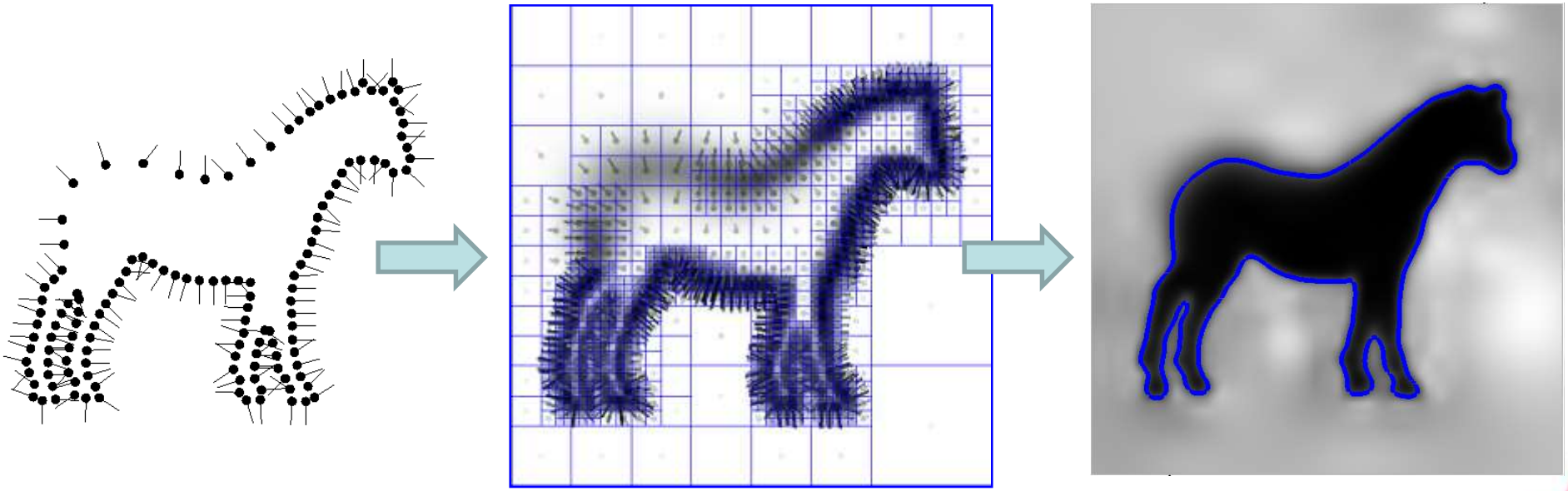


Oriented points **Indicator gradient**

[Kazhdan et al., 2006]

Algorithm:

1. Transform samples into a vector field.
2. Fit a scalar-field to the gradients.
3. Extract the isosurface.

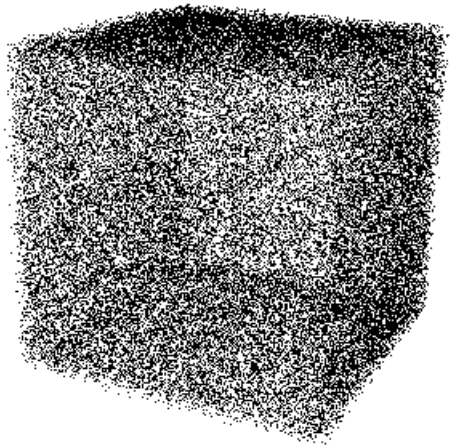


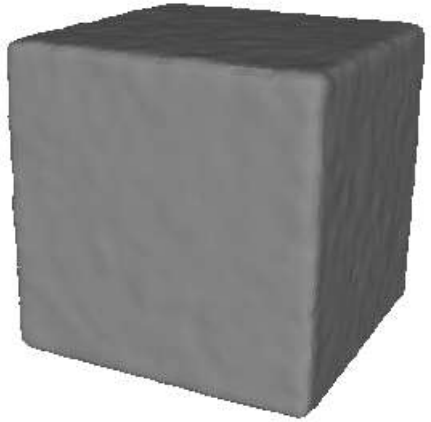
[Kazhdan et al., 2006]

Reconstruction of piecewise-planar objects

- Plane detection
- Plane assembling

Why Geometric primitives can be interesting for surface reconstruction ?



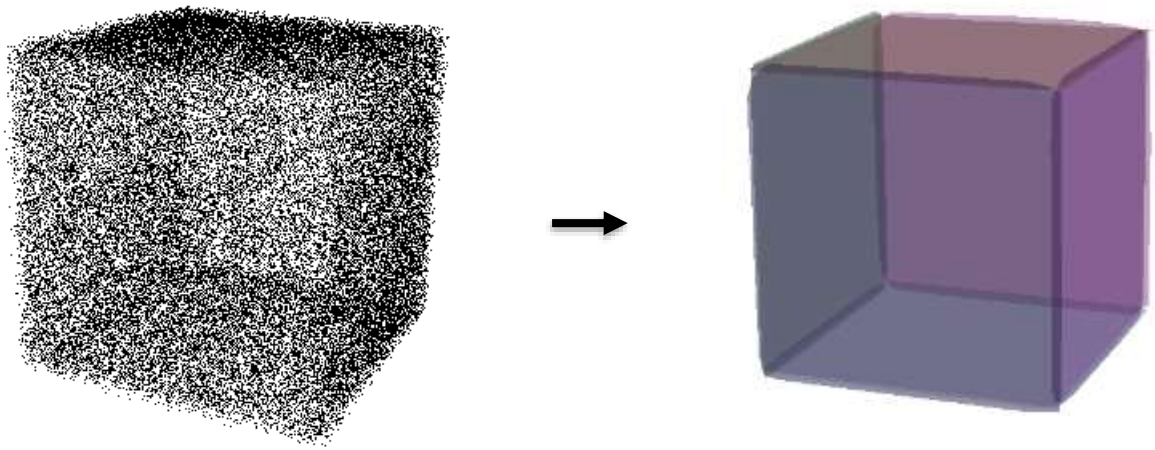


High complexity

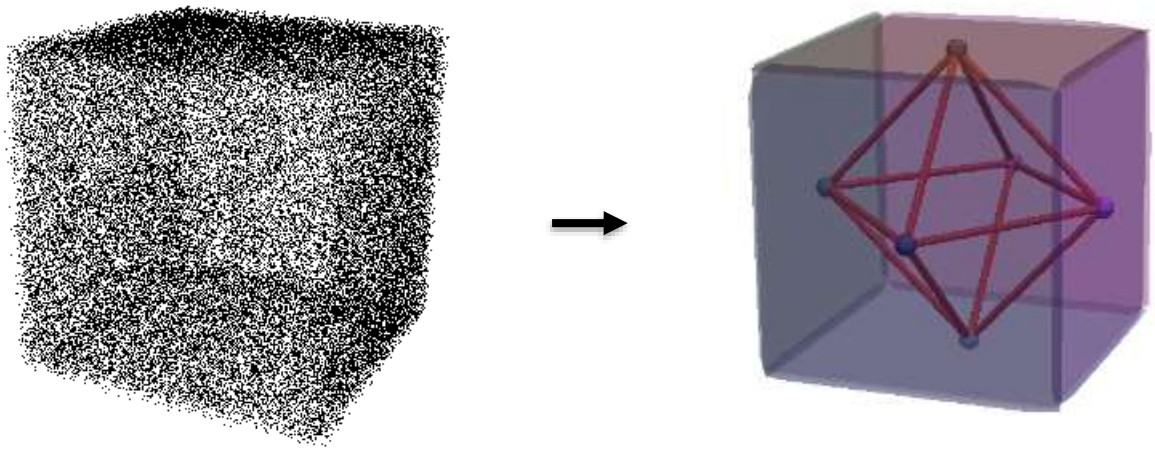
No structure

Smooth recontruction

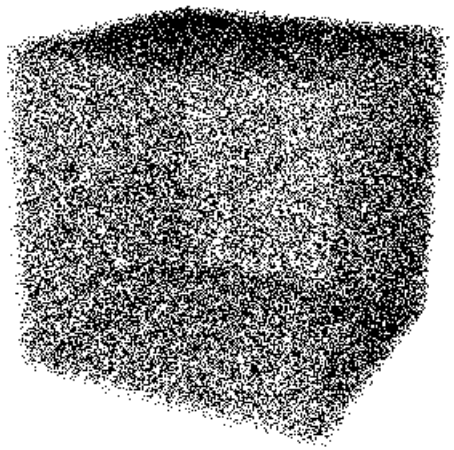
Why Geometric primitives can be interesting for surface reconstruction ?

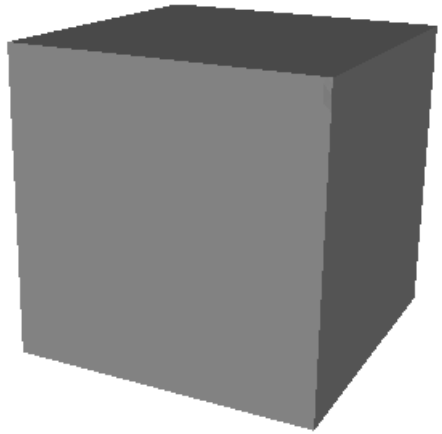


Why Geometric primitives can be interesting for surface reconstruction ?



Why Geometric primitives can be interesting for surface reconstruction ?

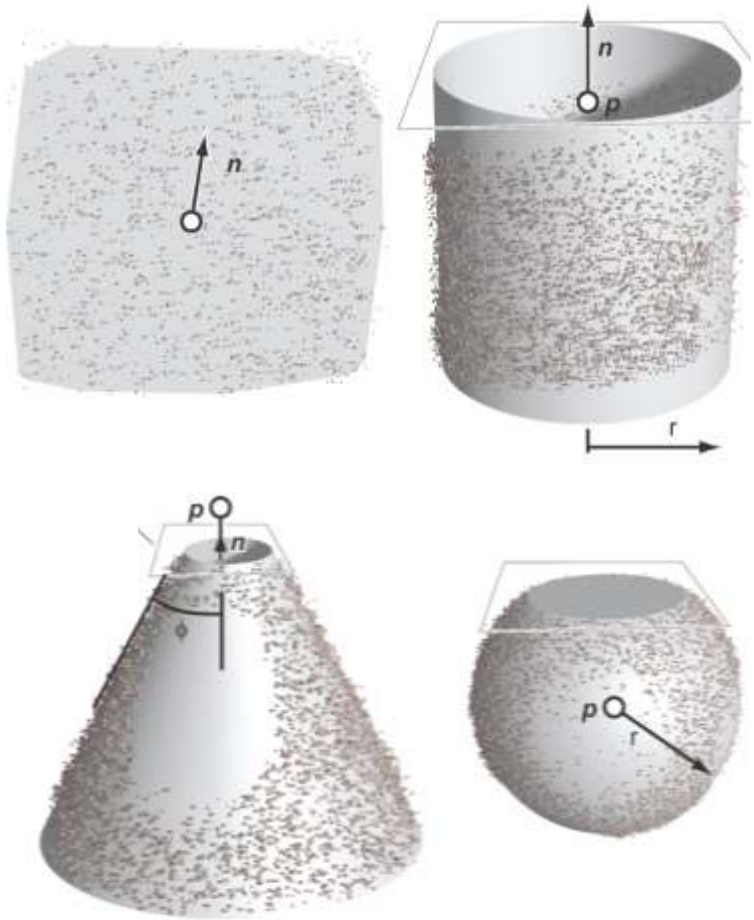


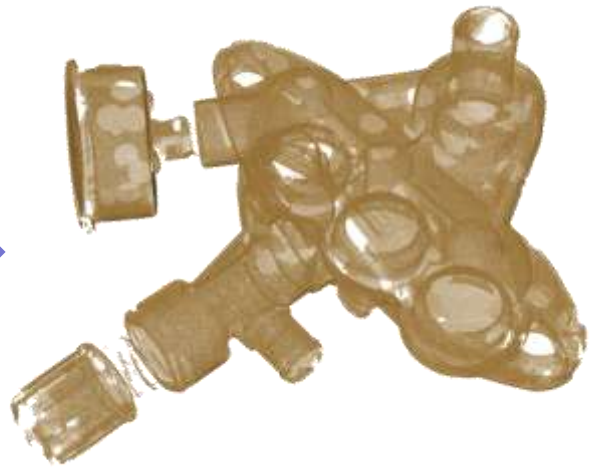


low complexity

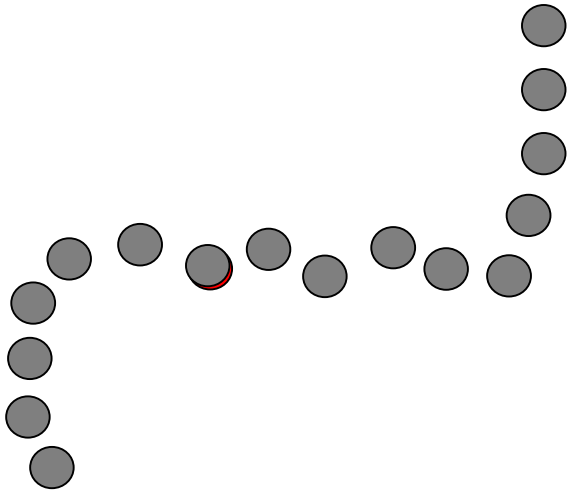
structure

How to extract Geometric primitives from point sets ?





Region growing



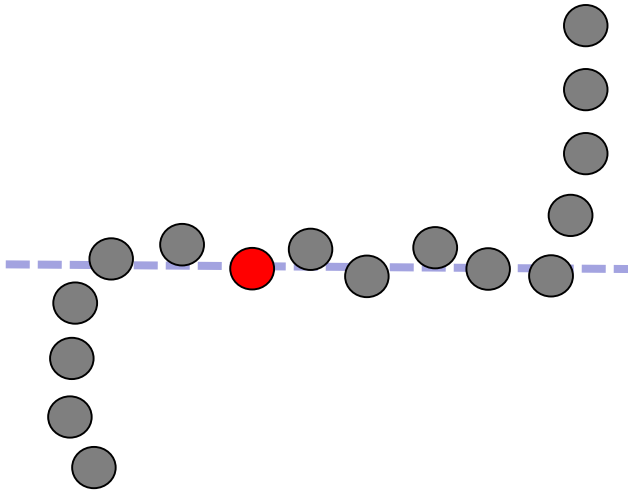
- Iterative method
- Spatial propagation of a primitive

Hypothesis

- deterministic
- Efficient for relatively “clean” Data

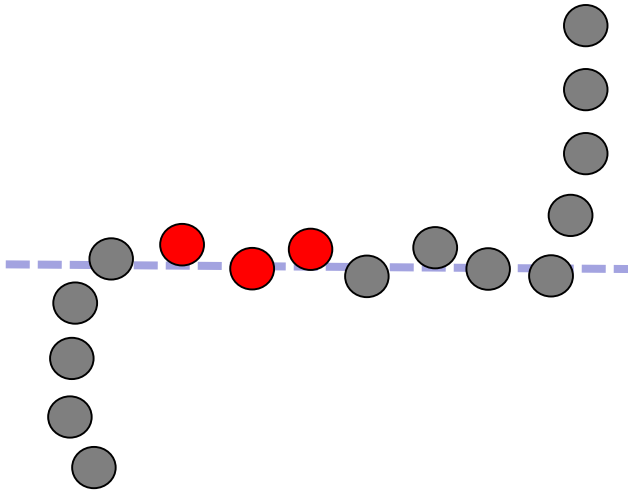
Region growing

- select a point and a primitive hypothesis

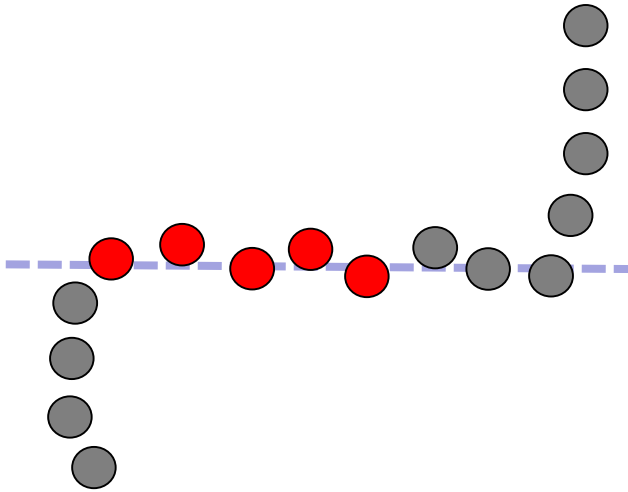


Region growing

- select a point and a primitive hypothesis
- propagate to the neighbors if they verify the hypothesis

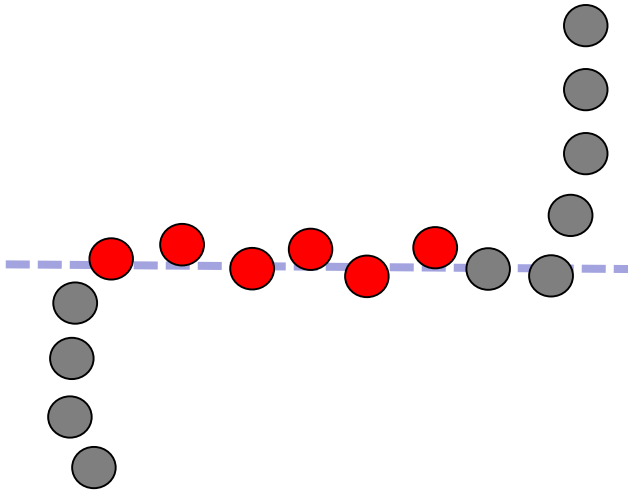


Region growing



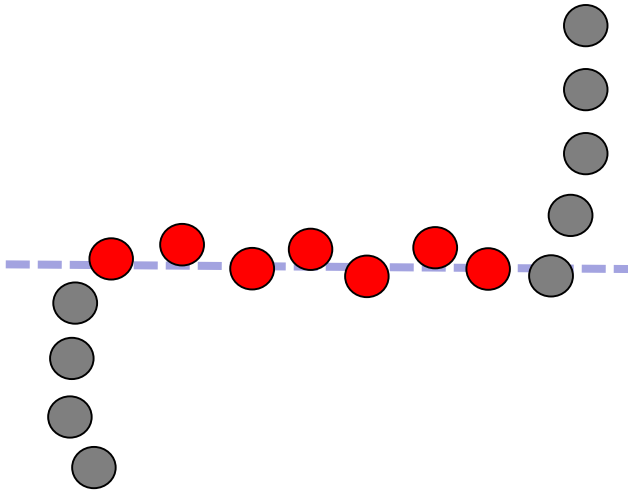
- select a point and a primitive hypothesis
- propagate to the neighbors if they verify the hypothesis, and iterate until no point verifies the hypothesis anymore.

Region growing



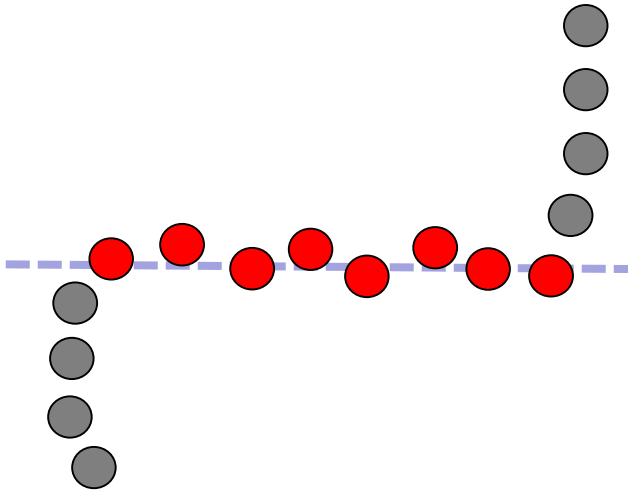
- select a point and a primitive hypothesis
- propagate to the neighbors if they verify the hypothesis, and iterate until no point verifies the hypothesis anymore.

Region growing



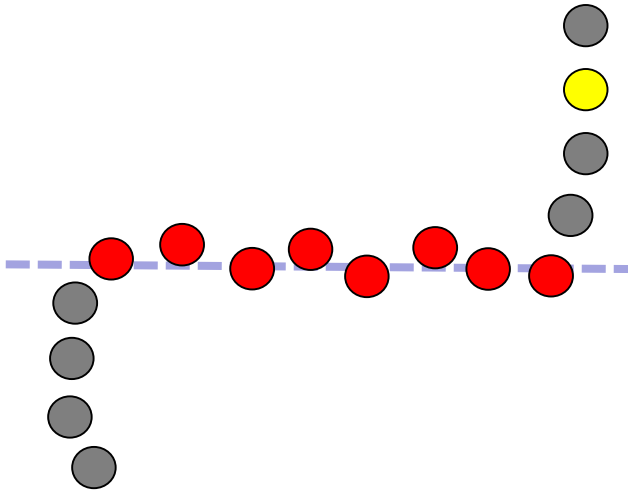
- select a point and a primitive hypothesis
- propagate to the neighbors if they verify the hypothesis, and iterate until no point verifies the hypothesis anymore.

Region growing



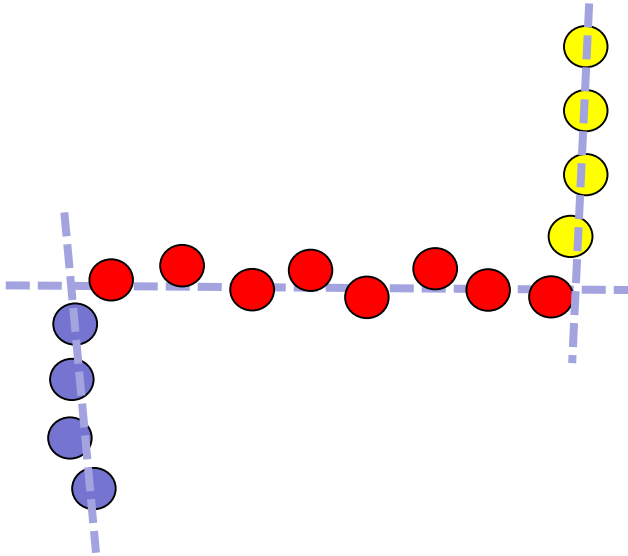
- select a point and a primitive hypothesis
- propagate to the neighbors if they verify the hypothesis, and iterate until no point verifies the hypothesis anymore.

Region growing



- select a point and a primitive hypothesis
- propagate to the neighbors if they verify the hypothesis, and iterate until no point verifies the hypothesis anymore.
- select a remaining point and a primitive Hypothesis, and iterate

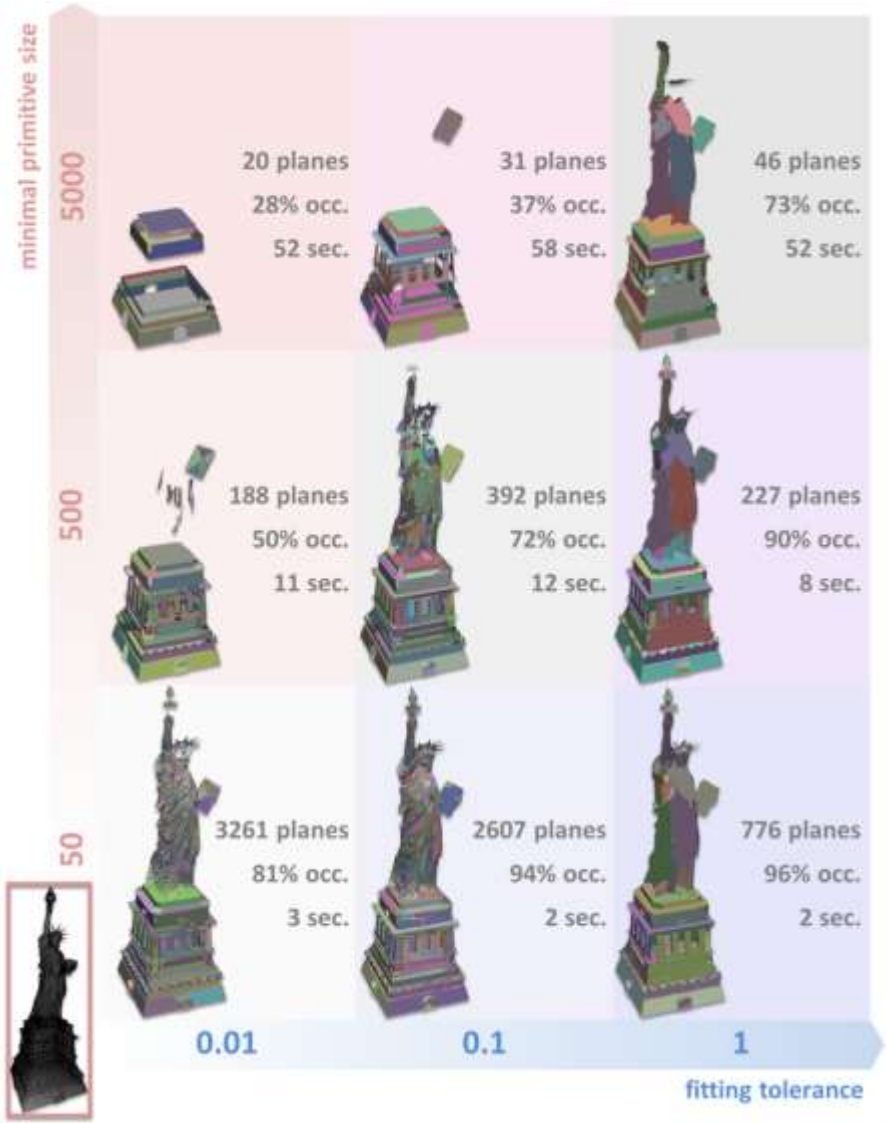
Region growing



- select a point and a primitive hypothesis
- propagate to the neighbors if they verify the hypothesis, and iterate until no point verifies the hypothesis anymore.
- select a remaining point and a primitive Hypothesis, and iterate

the parameters to specify

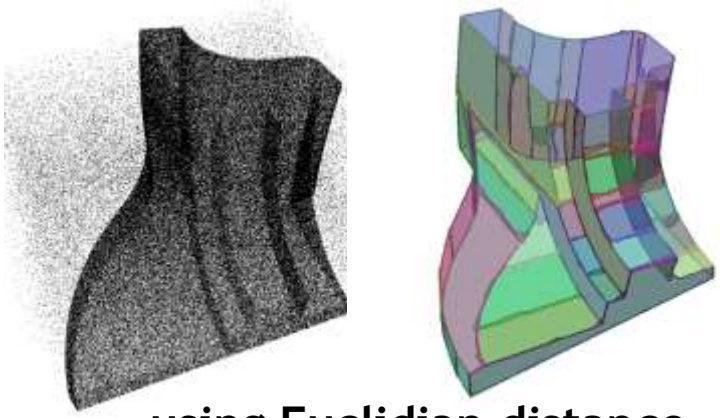
- minimum number of points needed to fit the primitive
- fitting tolerance



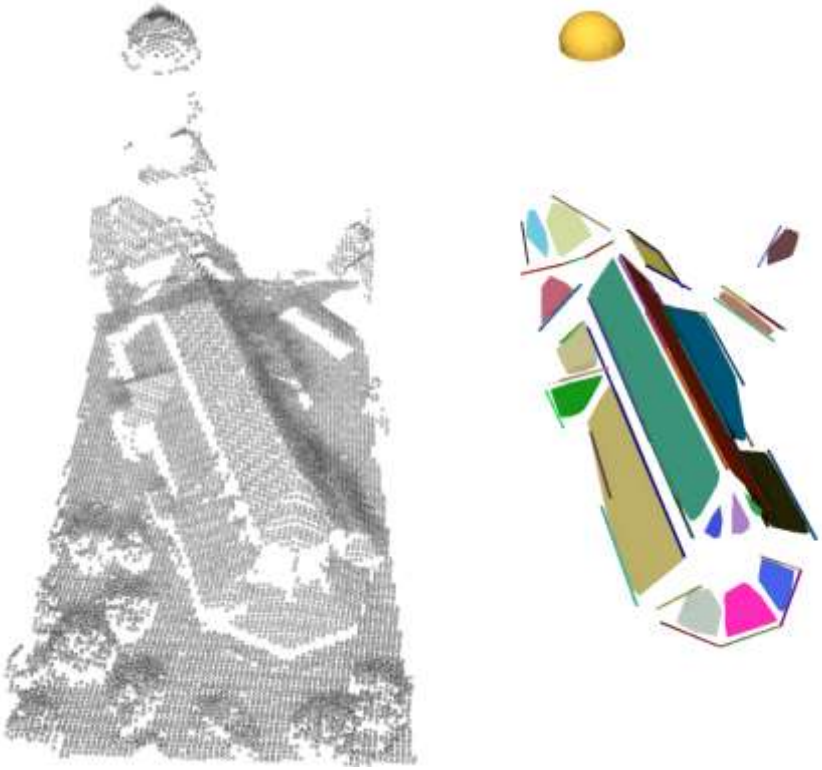
Region growing

- need to know the nearest neighbors
- the primitive hypothesis has to be relevant when starting the growing
- .. but the primitive hypothesis can also be updated during the growing
- not optimal when noisy data

Region growing



using Euclidian distance



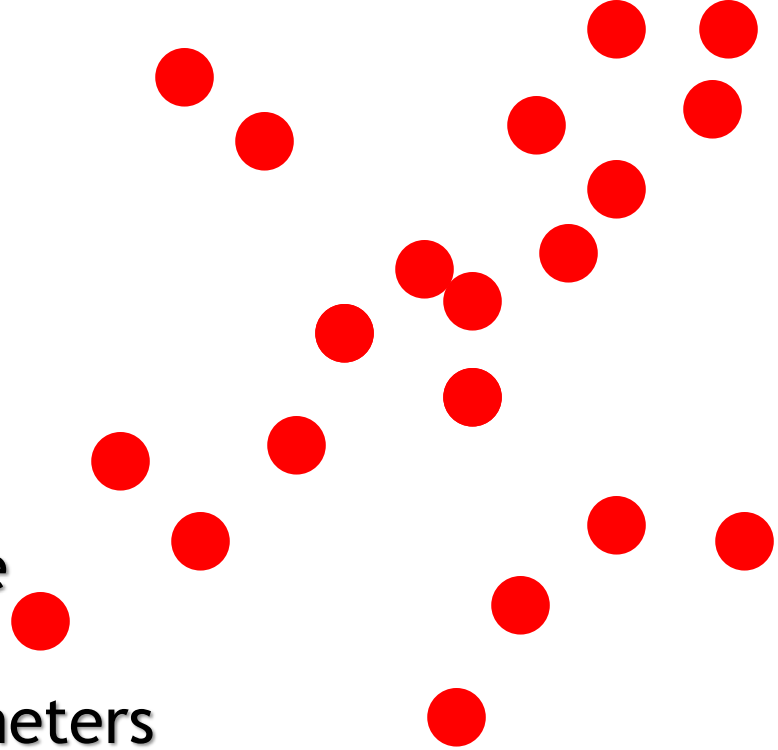
using normals

using normals and Euclidian distance

Ransac (RANdom SAmple Consensus)

- Iterative method
- Estimation of the primitive parameters by a random sampling of data
- Designed to be efficient with outlier-laden Data
- Non-deterministic

Ransac Algorithm



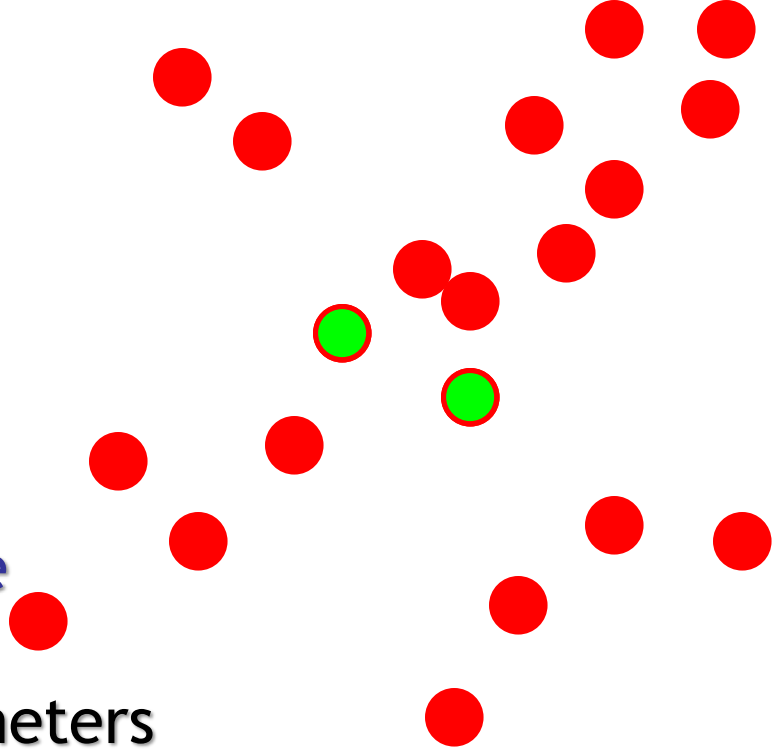
- Sample (randomly) the number of points required to fit the primitive

- Solve for primitive parameters using samples

- Score by the fraction of inliers within a preset threshold of the primitive

Repeat these 3 steps until the best primitive is found with high confidence

Ransac Algorithm



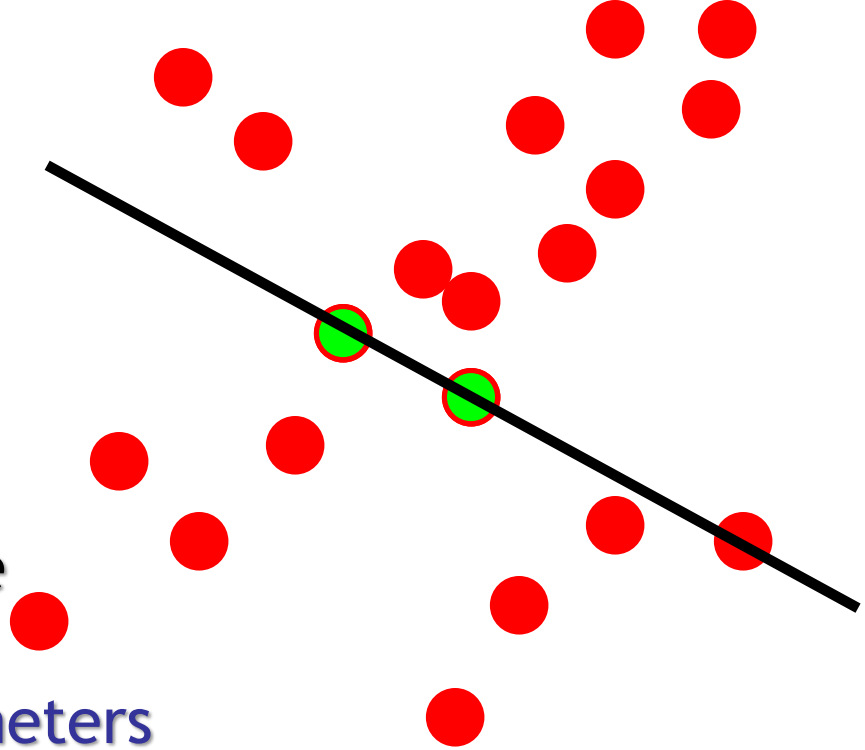
- Sample (randomly) the number of points required to fit the primitive

- Solve for primitive parameters using samples

- Score by the fraction of inliers within a preset threshold of the primitive

Repeat these 3 steps until the best primitive is found with high confidence

Ransac Algorithm



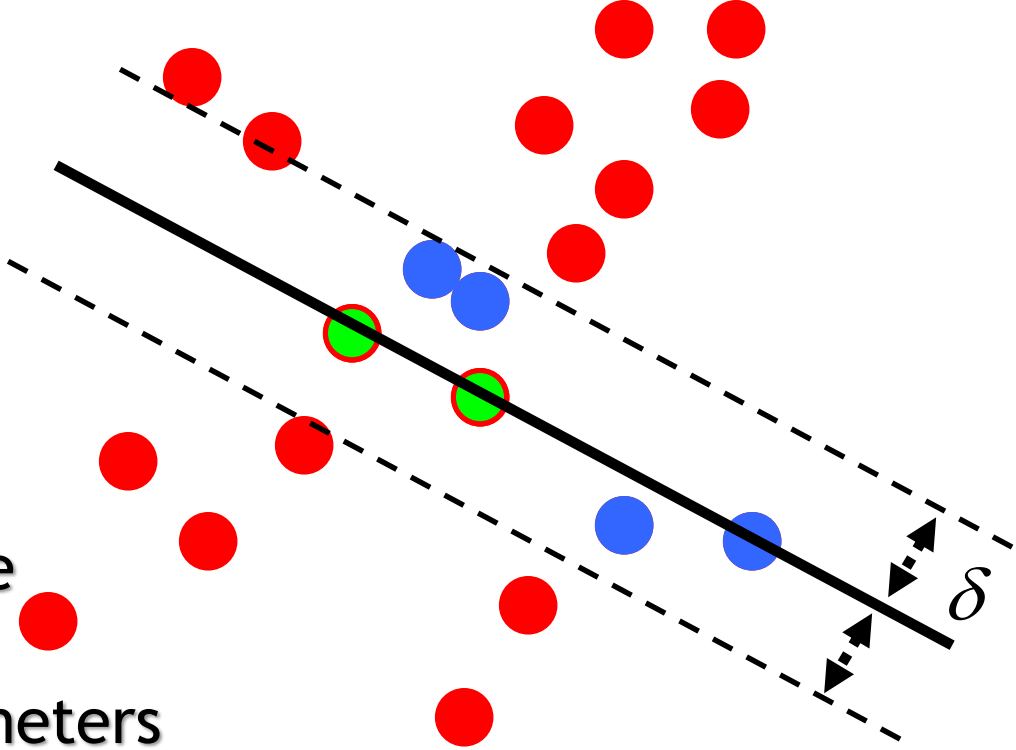
- Sample (randomly) the number of points required to fit the primitive

- Solve for primitive parameters using samples

- Score by the fraction of inliers within a preset threshold of the primitive

Repeat these 3 steps until the best primitive is found with high confidence

Ransac Algorithm



- Sample (randomly) the number of points required to fit the primitive

- Solve for primitive parameters using samples

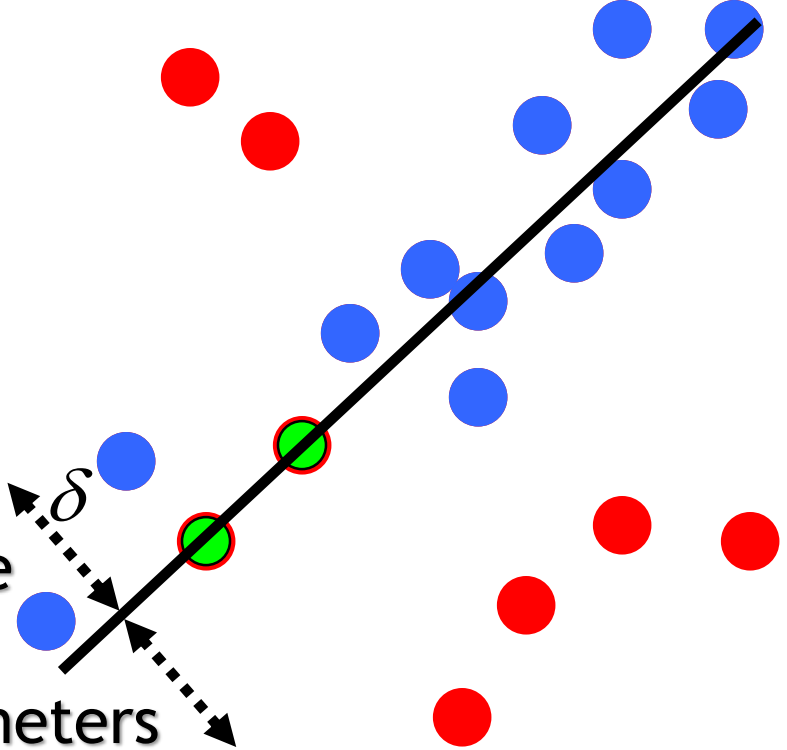
$$N_I = 6$$

- Score by the fraction of inliers within a preset threshold of the primitive

Repeat these 3 steps until the best primitive is found with high confidence

Ransac Algorithm

- Sample (randomly) the number of points required to fit the primitive



- Solve for primitive parameters using samples

$$N_I = 14$$

- Score by the fraction of inliers within a preset threshold of the primitive

Repeat these 3 steps until the best primitive is found with high confidence

the parameters to specify

- minimum number of points needed to fit the primitive
- Distance threshold δ
- Number of samples

To be chosen so that at least one random sample is free from outliers with a certain probability

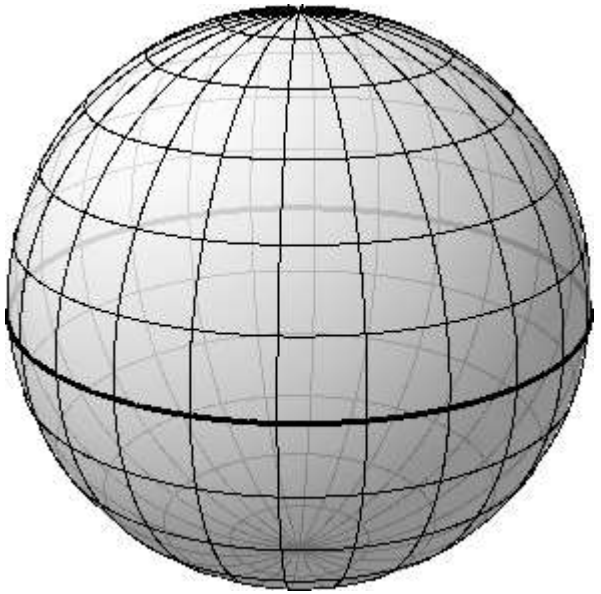
Accumulation methods

- Accumulate local primitive hypotheses in a space of primitive parameters

- extract the local maxima from the parameter space

- the parameter space must be discretized

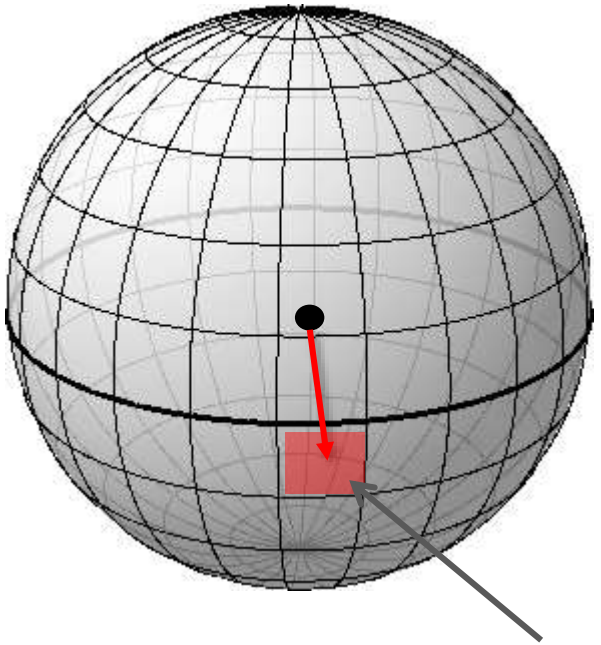
Accumulation methods: Gaussian sphere



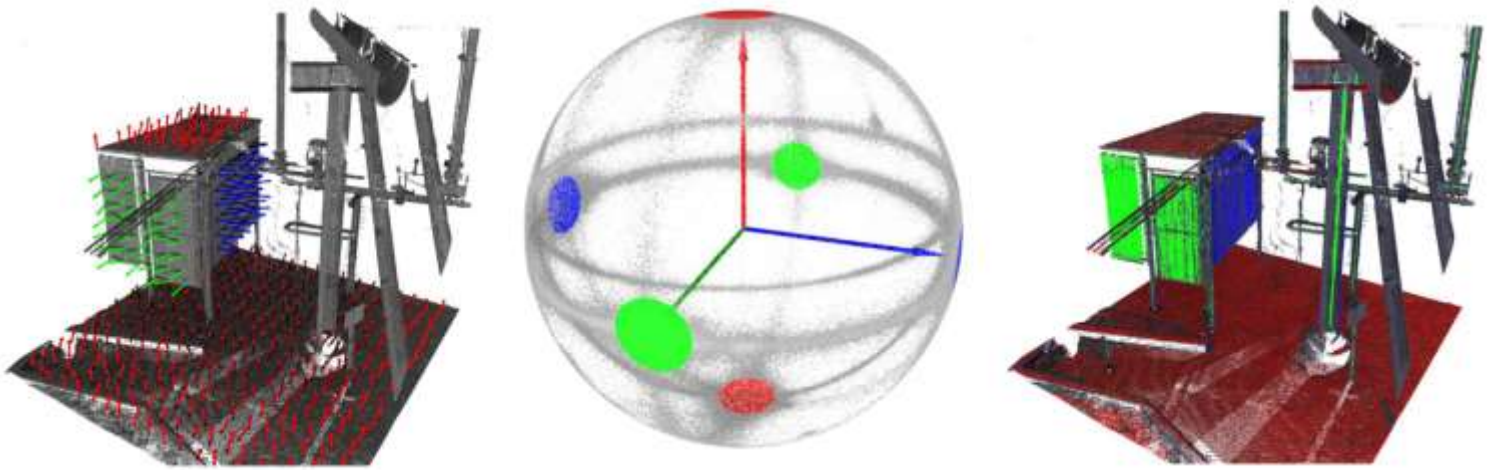
For each point of the data, we increment the sphere cell targeted by the point normal from the sphere center

Accumulation methods: Gaussian sphere

For each point of the data, we increment the sphere cell targeted by the point normal from the sphere center

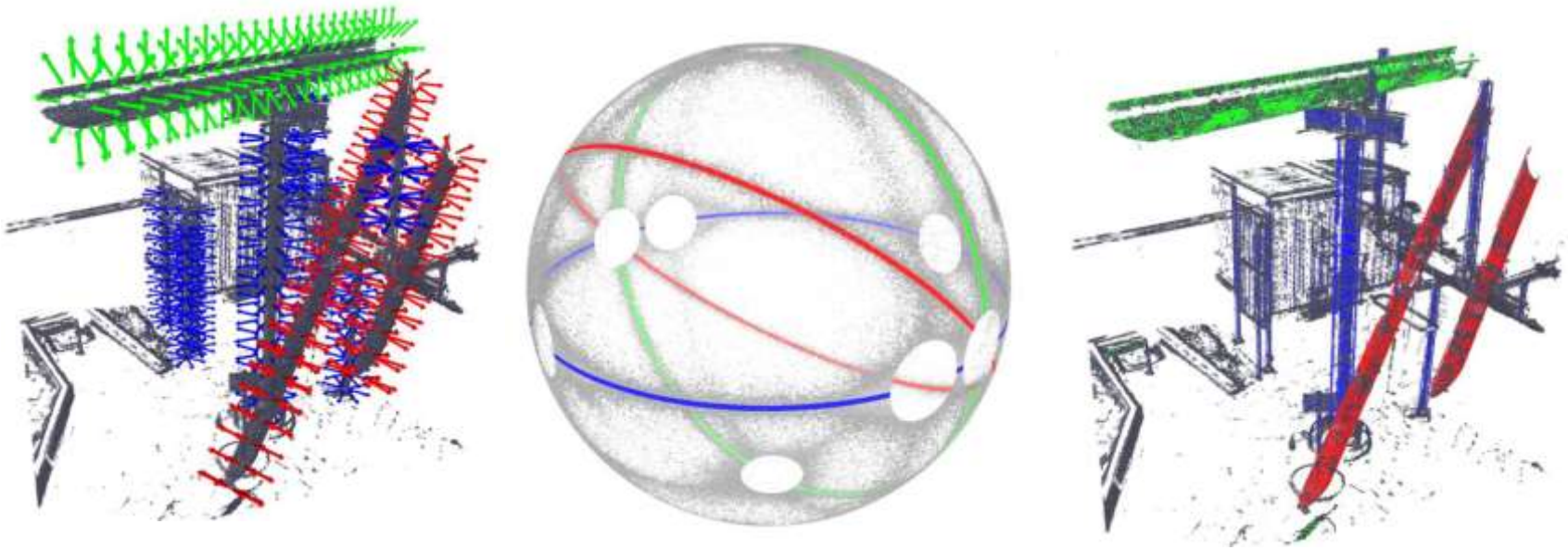


Accumulation methods: Gaussian sphere



An accumulation of points in the Gaussian sphere allows the detection of one or several planes with a similar orientation

Accumulation methods: Gaussian sphere



An accumulation of points along a circle in the Gaussian sphere allows the detection of one or several cylinders with a similar orientation

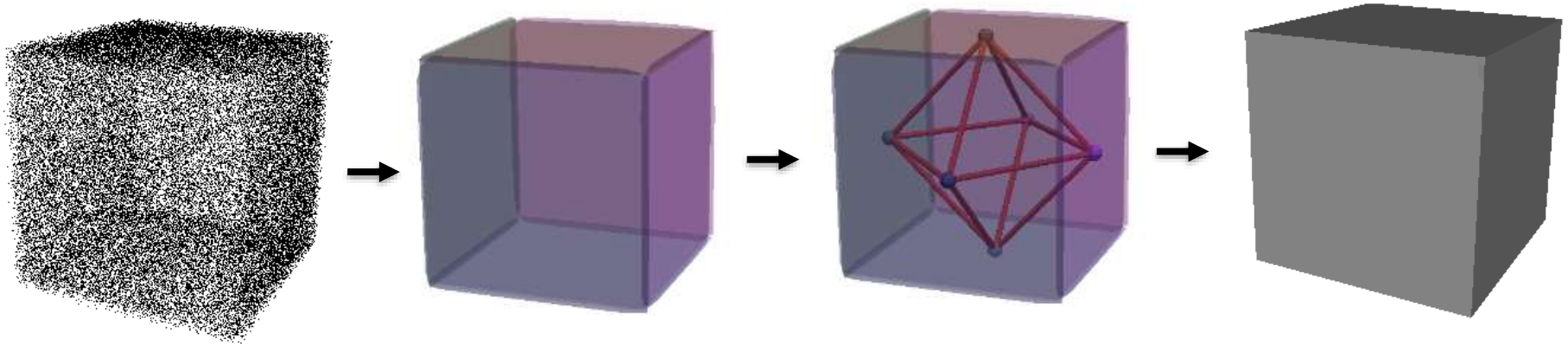
Accumulation methods

- can be computationally expensive
- restricted to certain types of primitives
- can be interesting for “structuring” the primitive configuration with global regularities

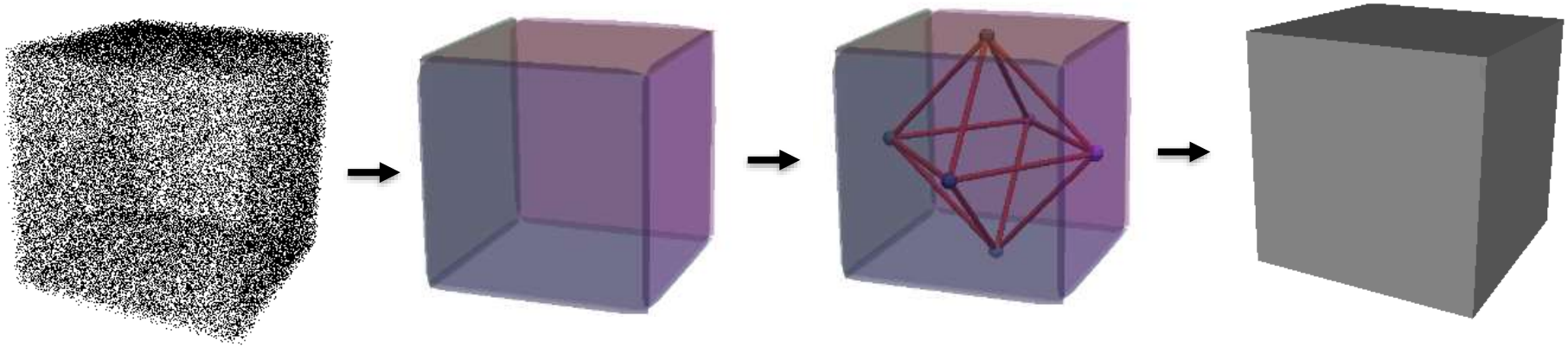
Surface reconstruction from geometric primitives

Q: What can we do once we have extracted the primitives ?

A1: compute the primitive adjacency graph, and reconstruct the surface as the dual of this graph.

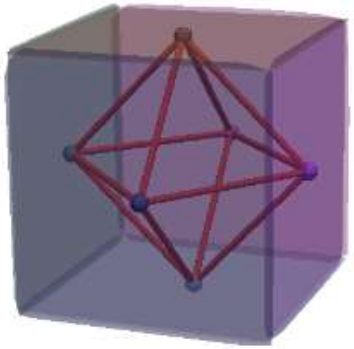


A1: compute the primitive adjacency graph, and reconstruct the surface as the dual of this graph.



If you are lucky..

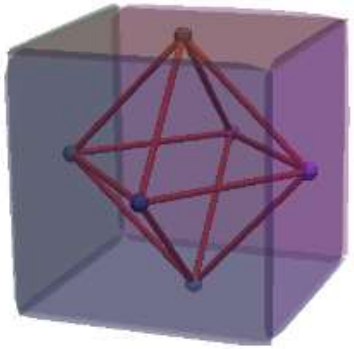
A1: compute the primitive adjacency graph, and reconstruct the surface as the dual of this graph.



Ideal case: this never happens in practice

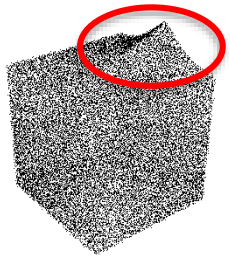
- No guarantee of finding the right primitive configuration and right adjacency graph

A1: compute the primitive adjacency graph, and reconstruct the surface as the dual of this graph.



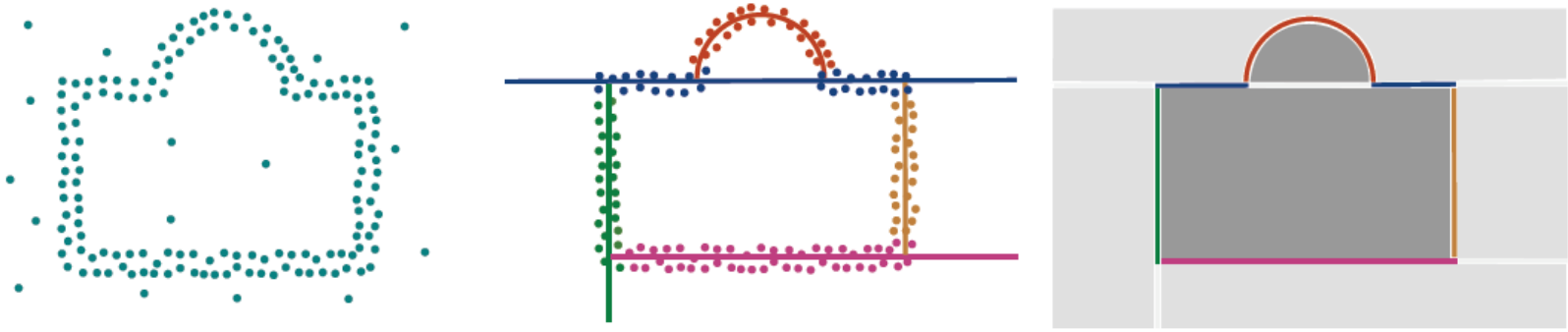
Ideal case: this never happens in practice

- No guarantee of finding the right primitive configuration and right adjacency graph



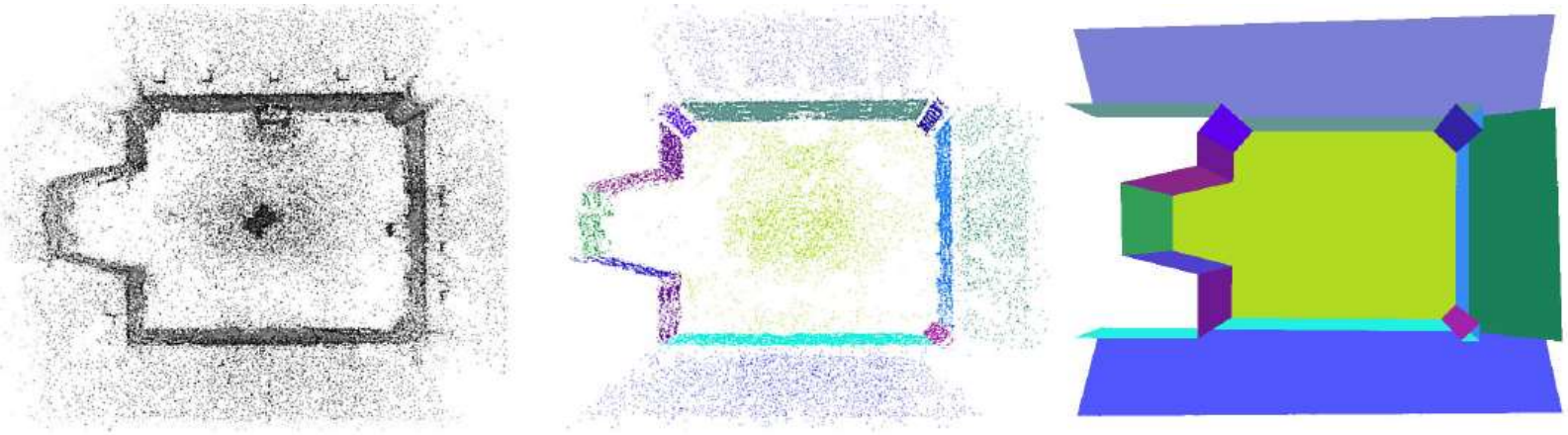
- No guarantee that the observed scene can be entirely explained by geometric primitives

- A2: Use primitives to partition the space into cells to be labeled as inside or outside



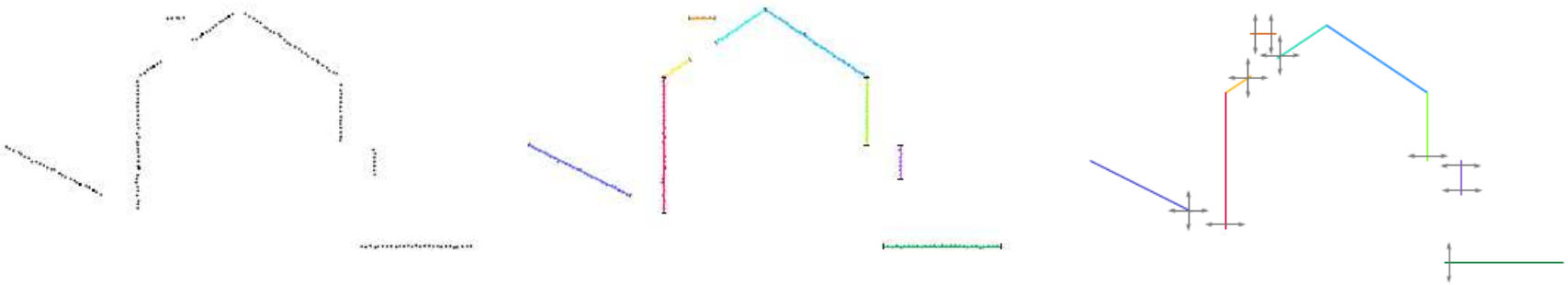
[Labatut et al., 2009]

- works well when no missing primitive

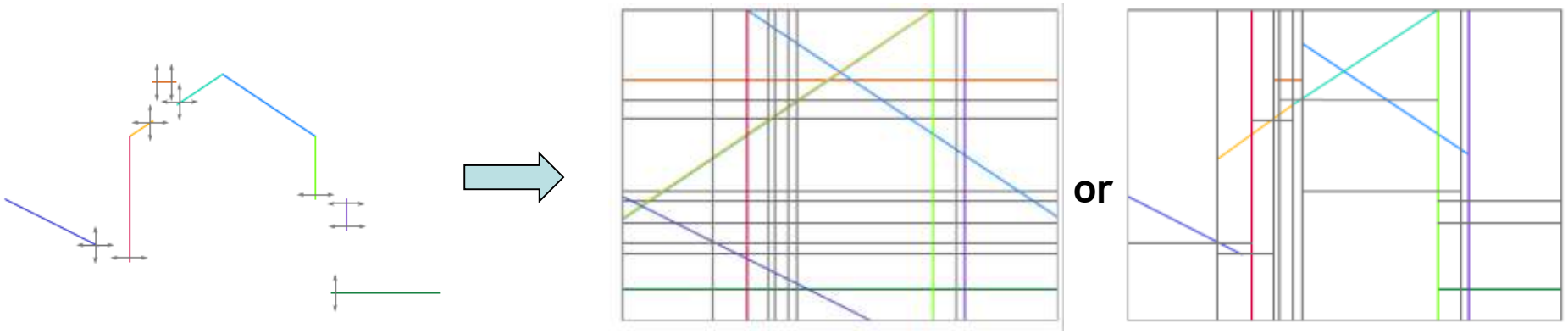


[Labatut et al., 2009]

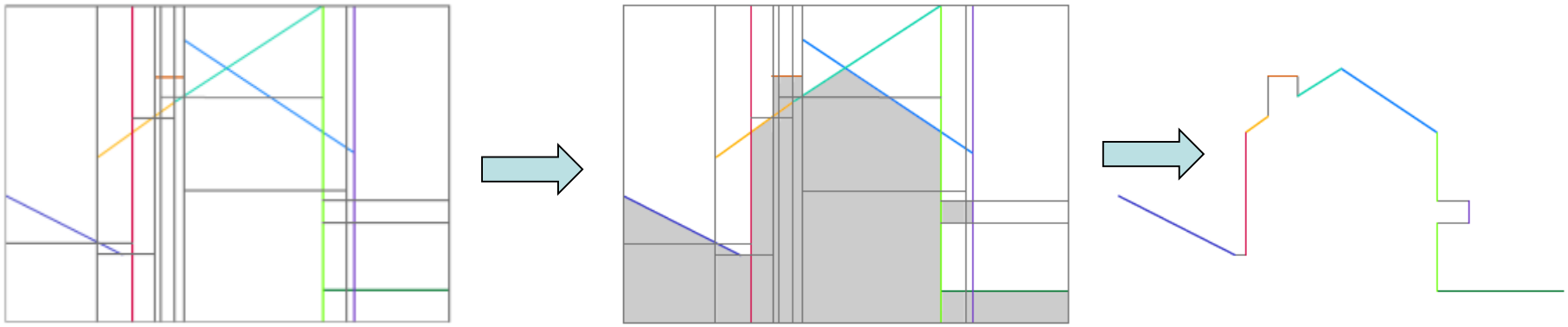
- when primitives are missed or cannot be detected, use of ghost primitives

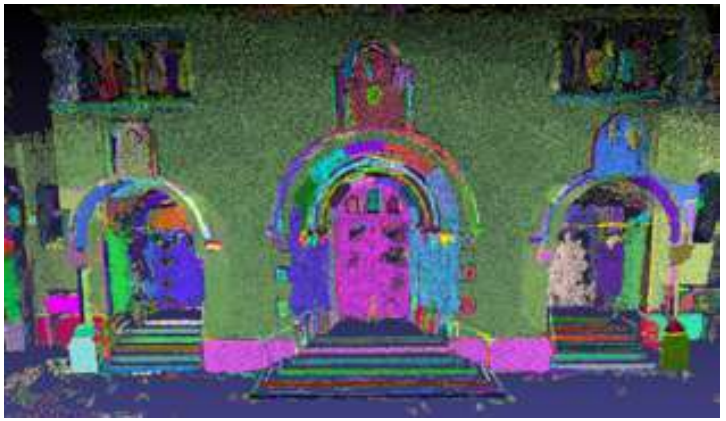


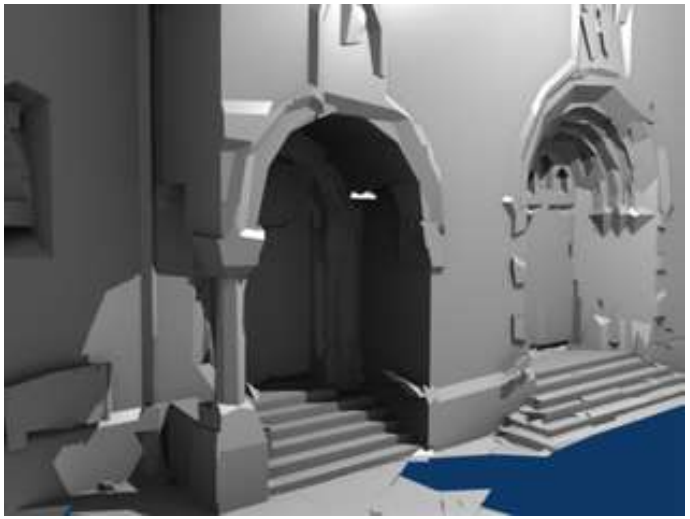
- when primitives are missed or cannot be detected, use of ghost primitives



- when primitives are missed or cannot be detected, use of ghost primitives







[Chauve et al., 2010]

City modeling

- Explicit methods
- Implicit methods

Urban objects

- Permanent elements: Buildings, roads, bridges, trees…
- Temporary elements: cars, fences, cranes…

Urban objects

Objects differ in terms of:

- density
- diversity
- dependence with each other

- Permanent elements: Buildings, roads, bridges, trees…
- Temporary elements: cars, fences, cranes…

Building reconstruction

Manhattan-world
[Furukawa et al., 2009] [Vanegas et al., 2010]
[Poullis et al., 2009] [Matei et al., 2008]

Piecewise planar structures
[Zebedin et al., 2008] [Brédif et al., 2007]

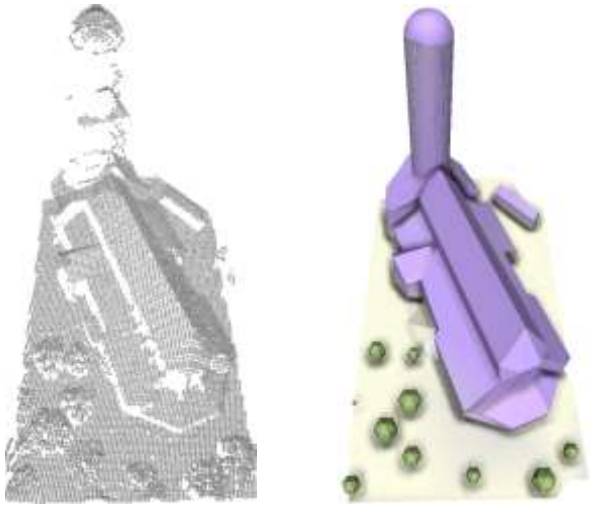
Block assembling / grammars
[Verma et al., 2006] [Lafarge et al., 2008]

Mesh simplification
[Zhou et al., 2010] [Verdié et al., 2011]

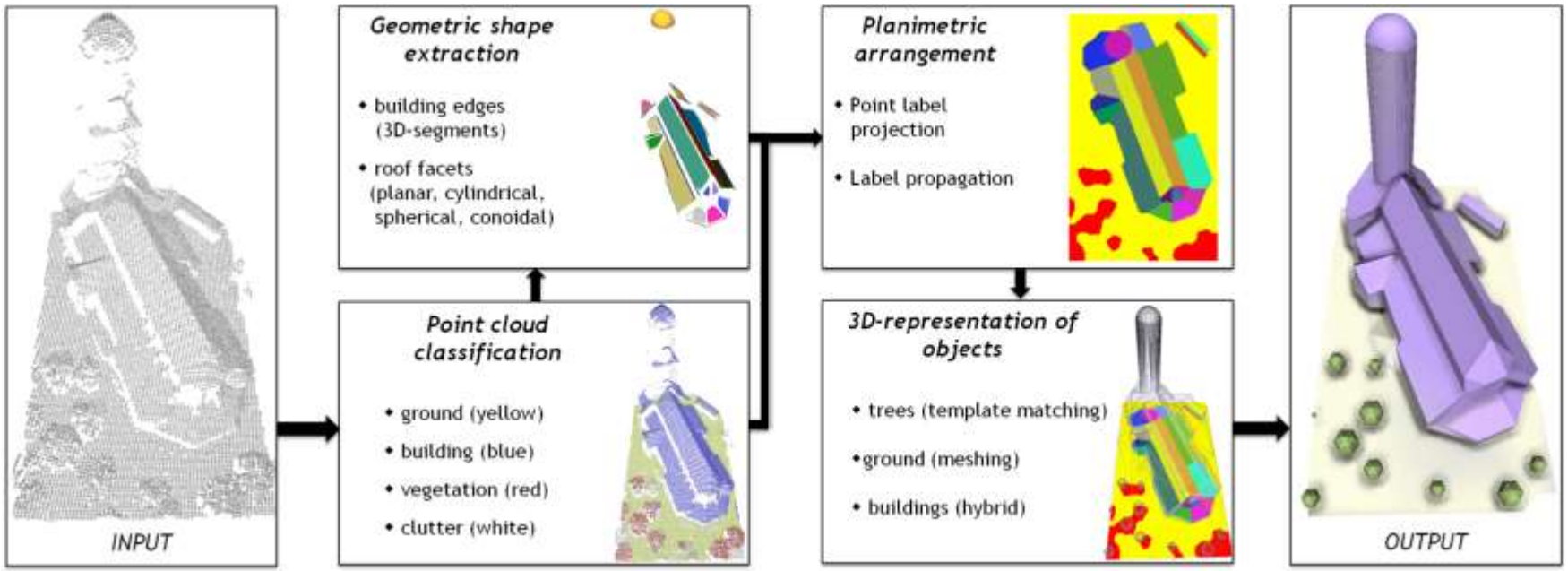
Compaction

Generality

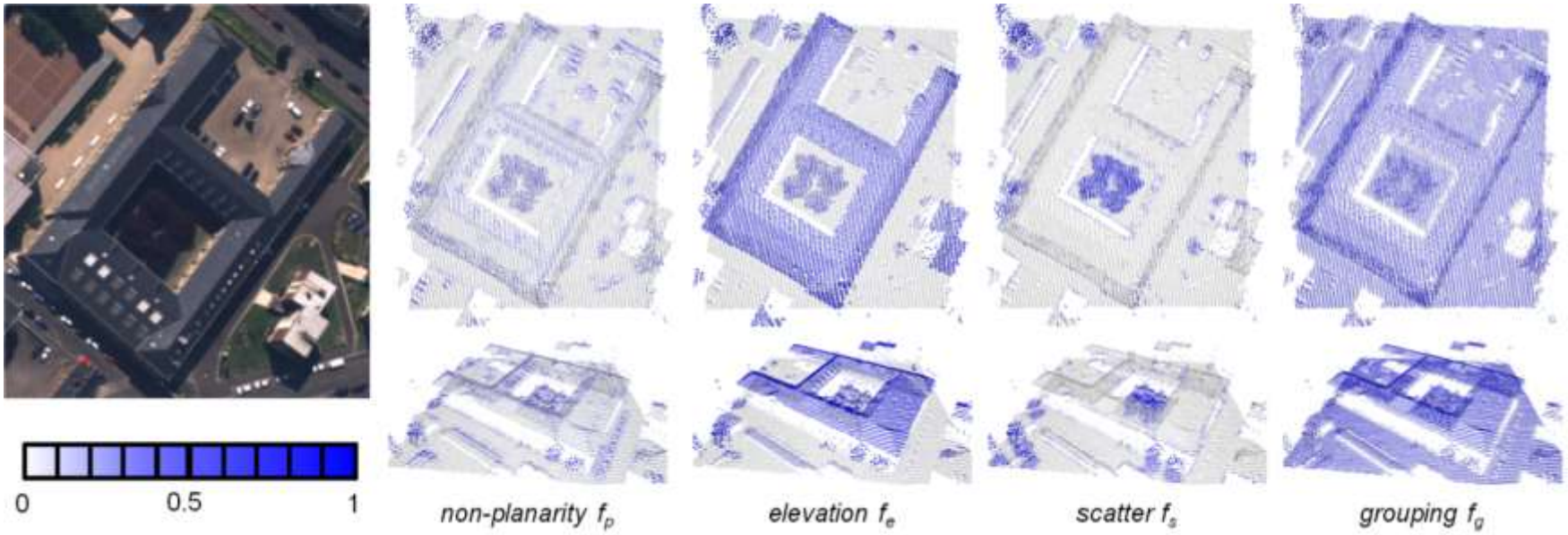
Example 1:
Reconstruction of cities from airborne Lidar



System overview



Discriminative attributes



classification

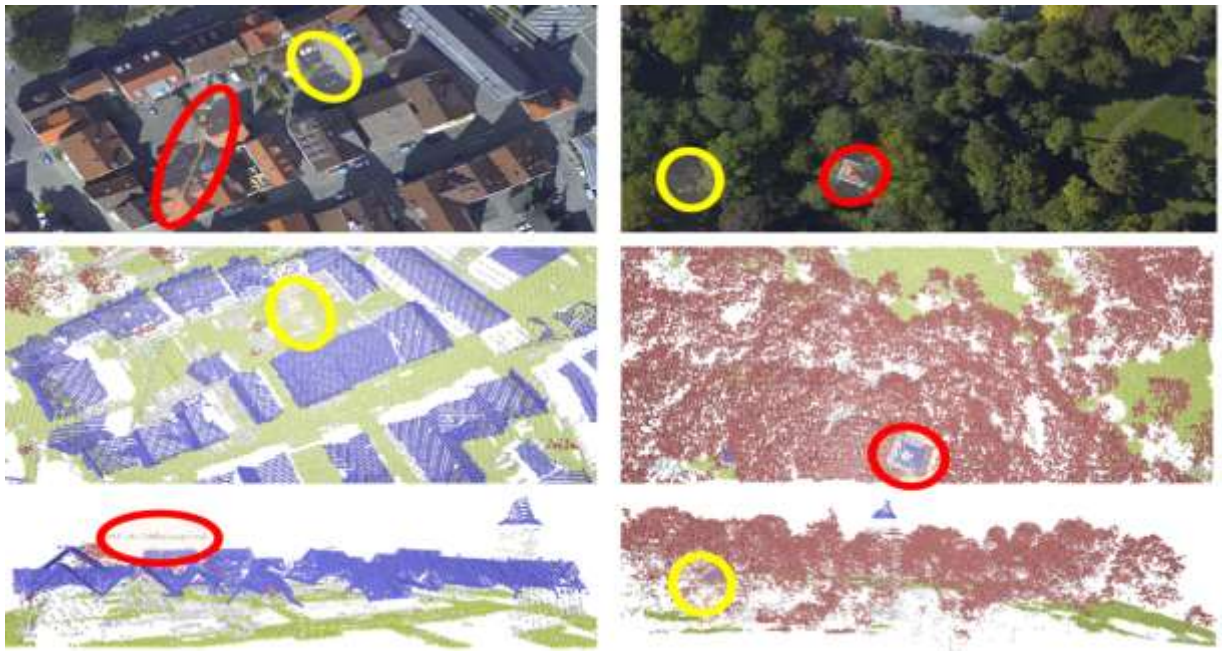
4 classes: building [blue], ground [yellow], vegetation [red], clutter [white]

Energy minimization: combination of the point descriptors + Potts model + Graph-Cuts

$$E(x) = \sum_{i=1..N_c} E_{di}(x_i) + \gamma \sum_{i \sim j} \mathbb{1}_{\{x_i \neq x_j\}}$$

$$E_{di}(x_i) = \begin{cases} (1-f_e).f_p.f_s & \text{if } x_i = building \\ (1-f_e).(1-f_p).(1-f_s) & \text{if } x_i = vegetation \\ f_e.f_p.f_s & \text{if } x_i = ground \\ (1-f_p).f_s.f_g & \text{if } x_i = clutter \end{cases}$$

classification

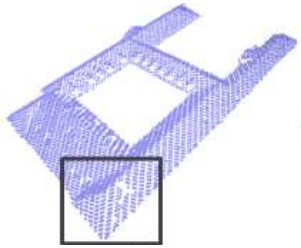


Color code: building [blue], ground [yellow], vegetation [red], clutter [white]

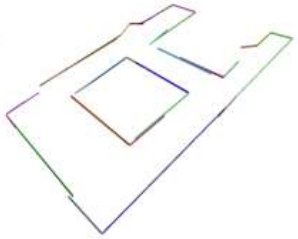
Clutter class includes

- outliers
- points of non significant urban objects (cars, wires, cranes, fences…)
- points of vertical structures (facades)

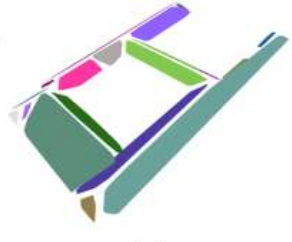
Primitive extraction



'*building*' points



3D-segments

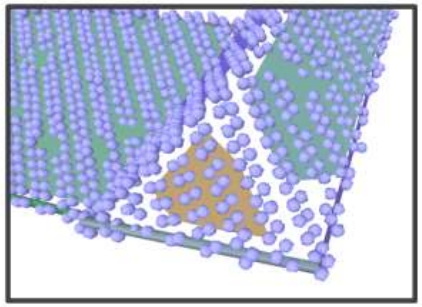


surface primitives

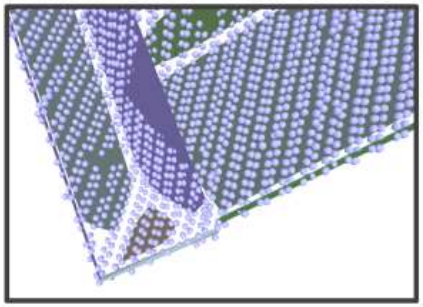
Building contours by 3D-segments

Roof sections

- by planes (region growing)
- by cylinders, spheres and cones on remaining points

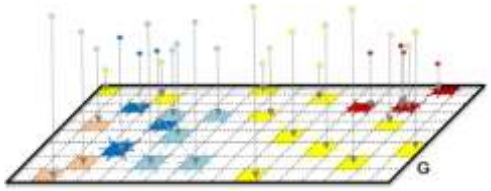


Crop: top view



Crop: bottom view

Planimetric labeling

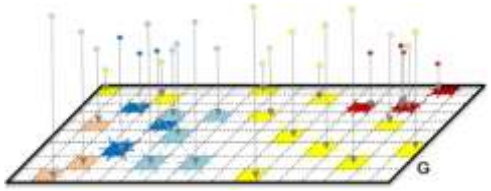


Configuration space L: point labels projected in a 2D-grid G

$$L = \{\text{ground}, \text{vegetation}, \text{plane}^{(l)}, \text{cylinder}^{(m)}, \text{sphere}^{(n)}, \text{cone}^{(o)}, \text{roof}\}^{card(G)}$$

Energy of standard form: $U(l) = \sum_{i \in G} D_i(l_i) + \beta \sum_{\{i,j\} \in E} V_{ij}(l_i, l_j)$

Planimetric labeling



Data term

$$D_i(l_i) = \begin{cases} c & \text{if } l_i = roof \\ \min(1, |z_{l_i} - z_{p_i}|) & \text{else if } i \in G^{(proj)} \\ 0 & \text{otherwise} \end{cases}$$

- altimetric error between the surface associated with l_i and the highest point of the cell *i*

- *c* controls the occurrence of irregular roof sections w.r.t. regular ones

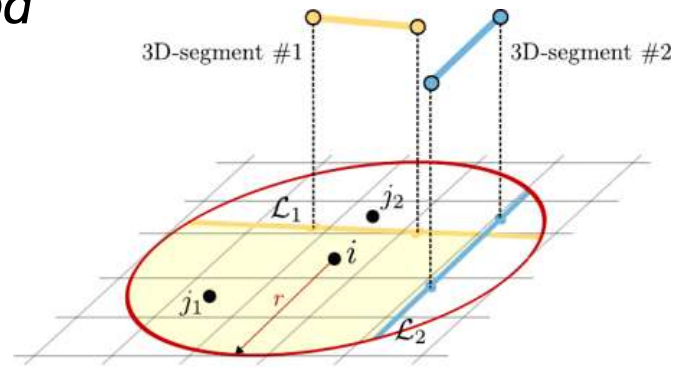
Planimetric labeling

Propagation constraints

$$V_{ij}(l_i, l_j) = \begin{cases} \epsilon_1 \text{ if } l_i \bowtie l_j \\ \epsilon_2 \text{ if } l_i = l_j \\ 1 \quad \text{otherwise} \end{cases}$$

- *breakline-dependent neighborhood*

$$\{i, j\} \in E \Leftrightarrow \begin{cases} \|i - j\|_2 \leq r \\ \mathcal{O}(i, \mathcal{L}_k) = \mathcal{O}(j, \mathcal{L}_k) \end{cases}$$



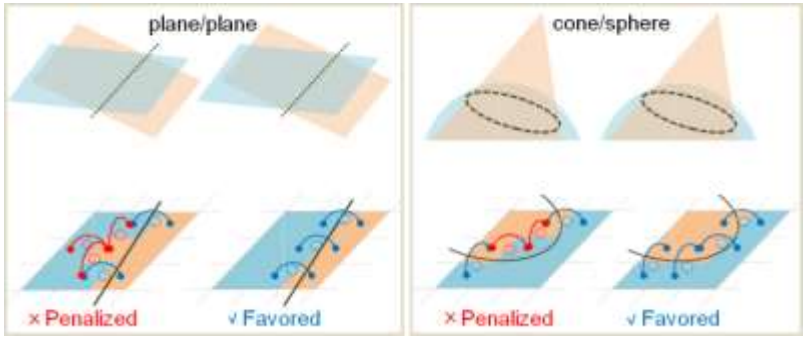
Planimetric labeling

Propagation constraints

$$V_{ij}(l_i, l_j) = \begin{cases} \epsilon_1 \text{ if } l_i \bowtie l_j \\ \epsilon_2 \text{ if } l_i = l_j \\ 1 \quad \text{otherwise} \end{cases}$$

- *breakline-dependent neighborhood*
- *structure arrangement law*

$$l_i \bowtie l_j \Leftrightarrow \mathcal{O}(i, \mathcal{I}_{l_i,l_j}) \neq \mathcal{O}(j, \mathcal{I}_{l_i,l_j})$$



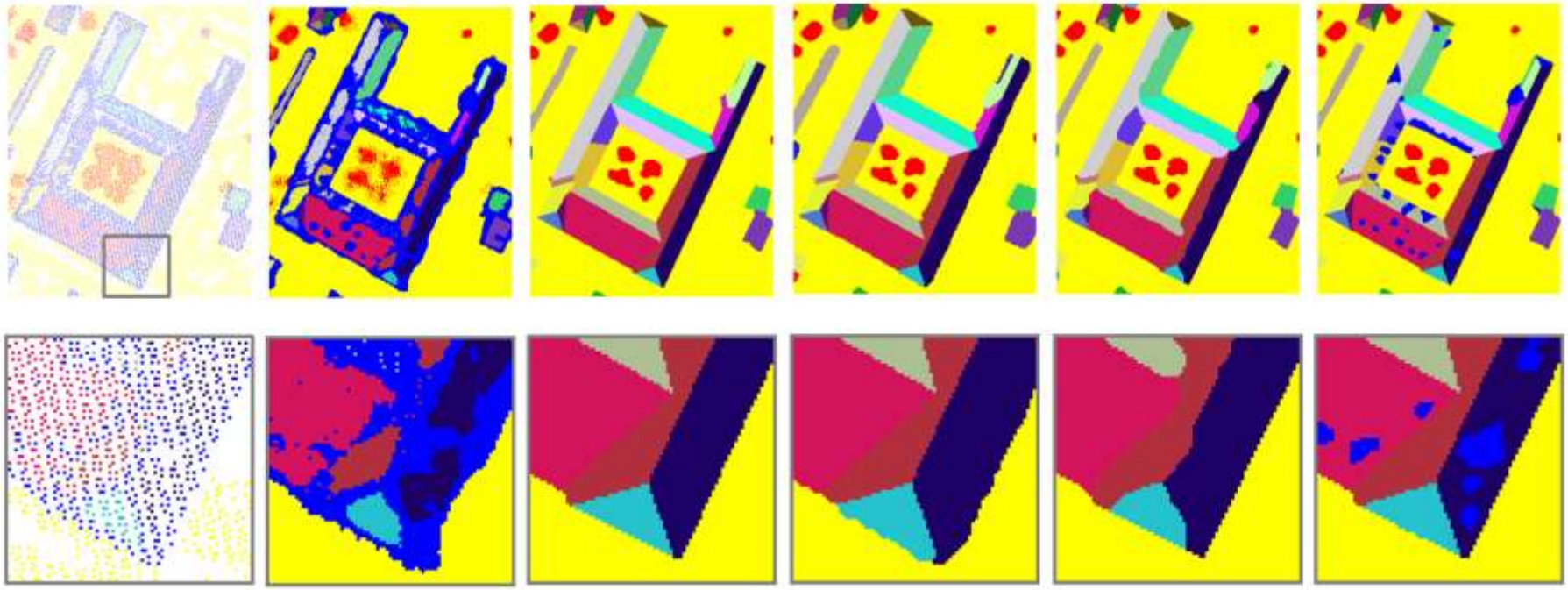
Planimetric labeling

Propagation constraints

$$V_{ij}(l_i, l_j) = \begin{cases} \epsilon_1 \text{ if } l_i \bowtie l_j \\ \epsilon_2 \text{ if } l_i = l_j \\ 1 \quad \text{otherwise} \end{cases}$$

- *breakline-dependent neighborhood*
- *structure arrangement law*
- *label smoothness*

Impact of the various energy composants



labels originally projected on the 2D-grid

initial label map

label map after minimizing U

label map after minimizing U without breakline-dependent neighborhood

label map after minimizing a variant of U without structure arrangement

label map after minimizing U whose parameter c has been significantly decreased

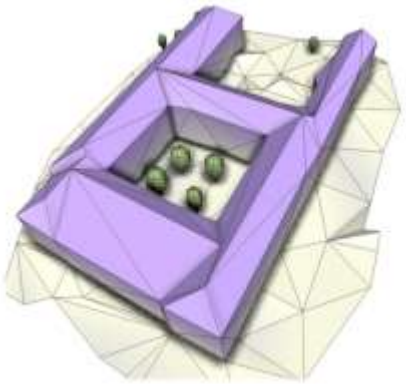
Color code: roof [blue], ground [yellow], vegetation [red], empty cell [white] , surface primitives [random color]

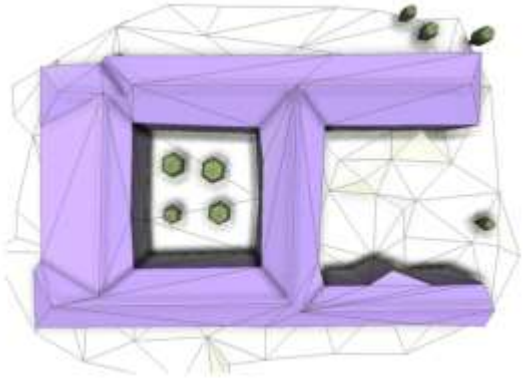
Object modeling

Buildings

- Hybrid repesentation

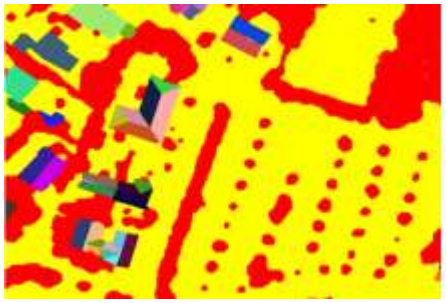
(mesh+3D-primitives)

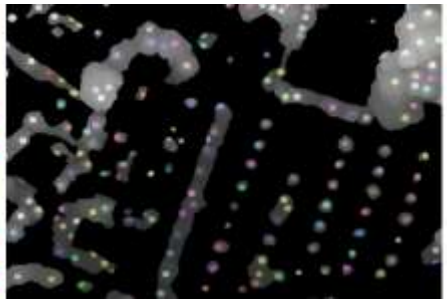


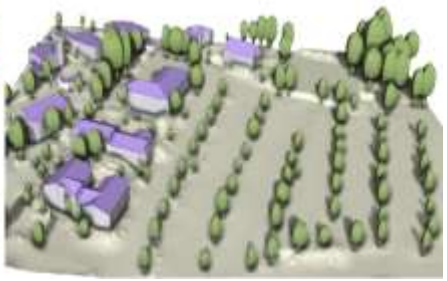


Trees

- Template matching (ellipsoid)





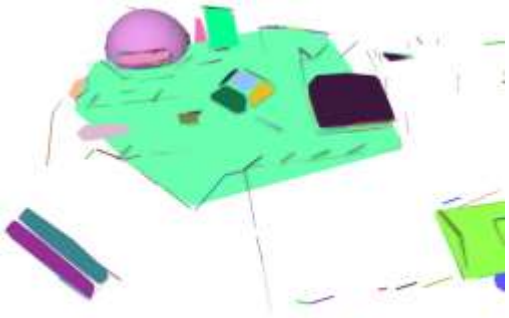


Ground

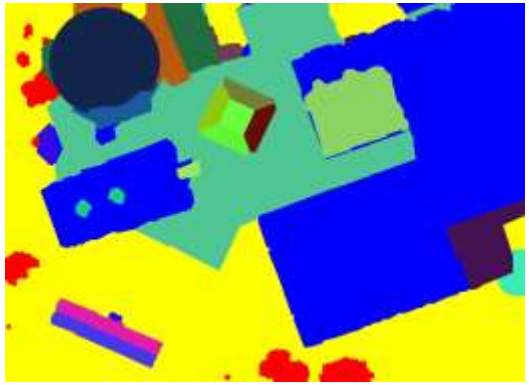
- mesh

Results

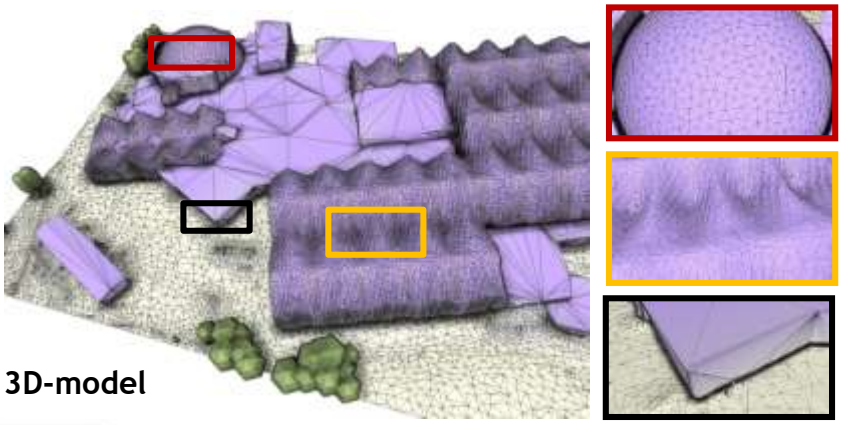
Aerial image



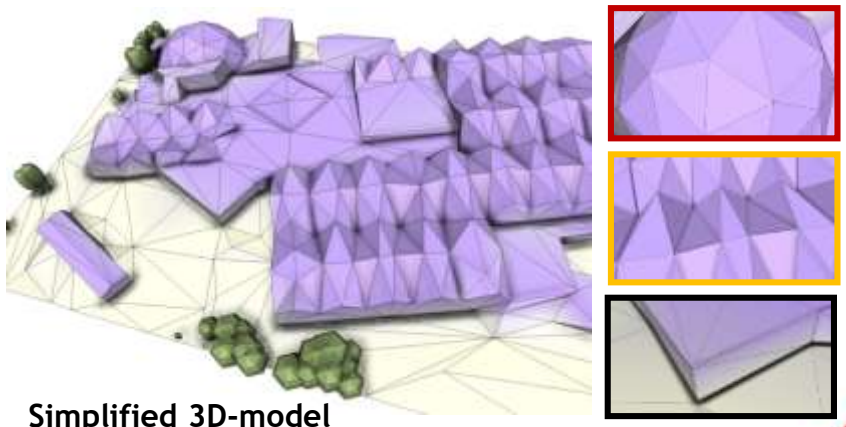
Extracted primitives



Label map



3D-model



Simplified 3D-model

Biberach, Germany (1km², 2.3M points)

Aerial image (from Google Maps)

Classified point cloud [color code: blue=building, red=vegetation, yellow=ground, white=clutter]

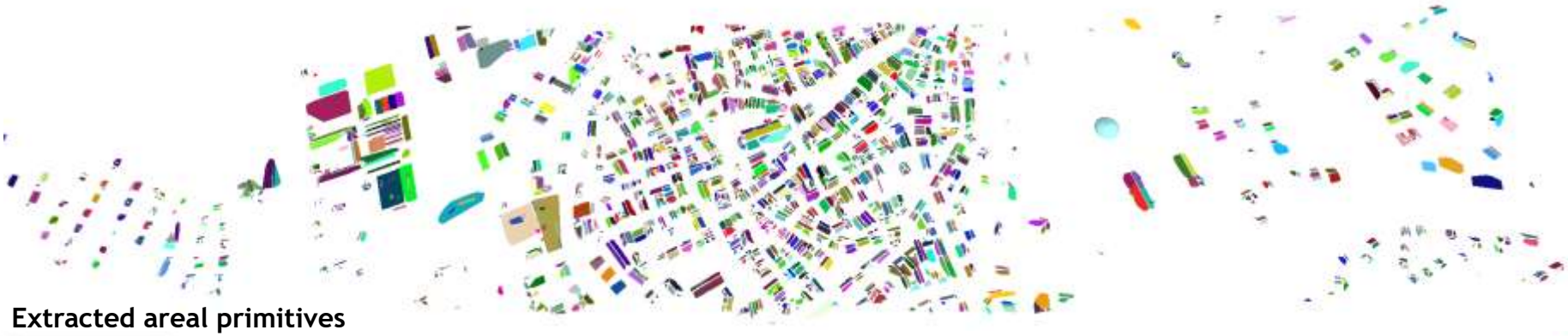
Biberach, Germany (1km², 2.3M points)

Aerial image (from Google Maps)

Extracted 3D-segments

Biberach, Germany (1km², 2.3M points)

Aerial image (from Google Maps)



Extracted areal primitives

Biberach, Germany (1km², 2.3M points)

Aerial image (from Google Maps)

Block decomposition

Biberach, Germany (1km², 2.3M points)

Aerial image (from Google Maps)

Label map

Biberach, Germany (1km², 2.3M points)

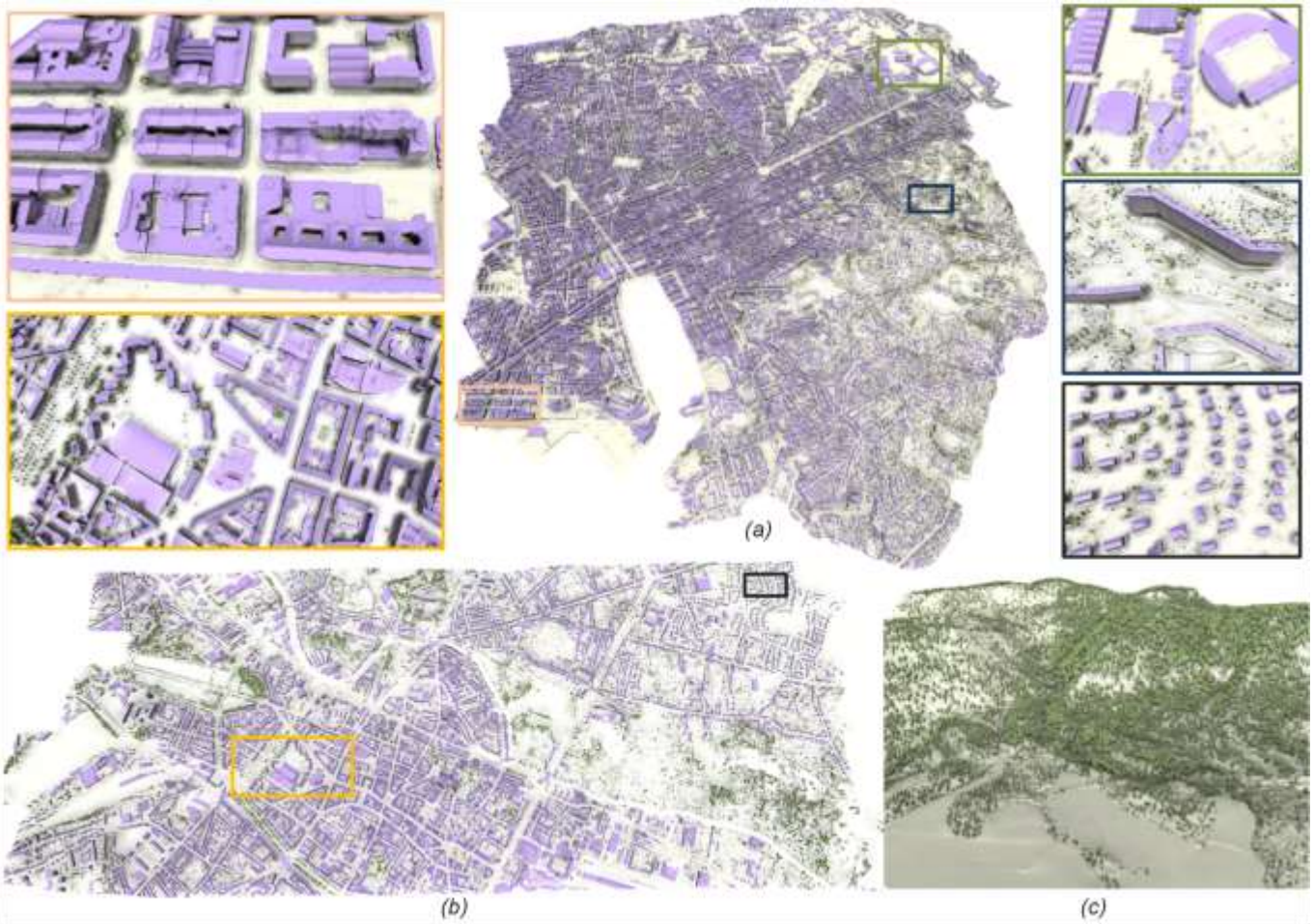
Aerial image (from Google Maps)

3D-model

Biberach, Germany (1km², 2.3M points)

Aerial image (from Google Maps)

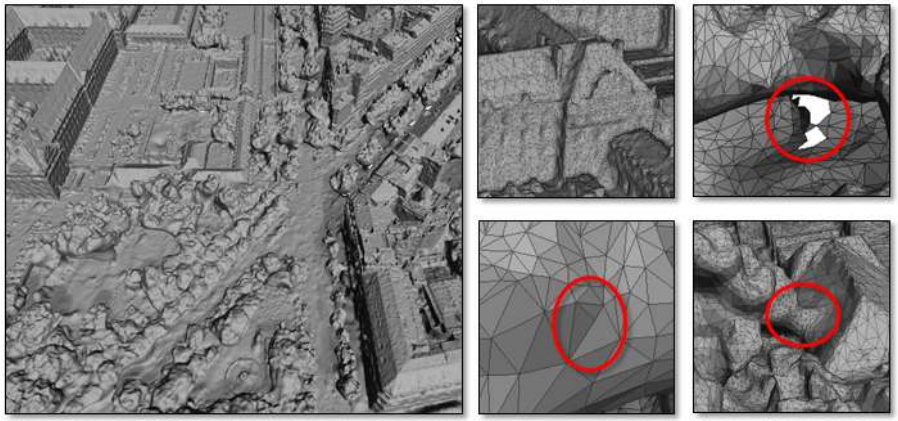
3D-model with mesh visualization



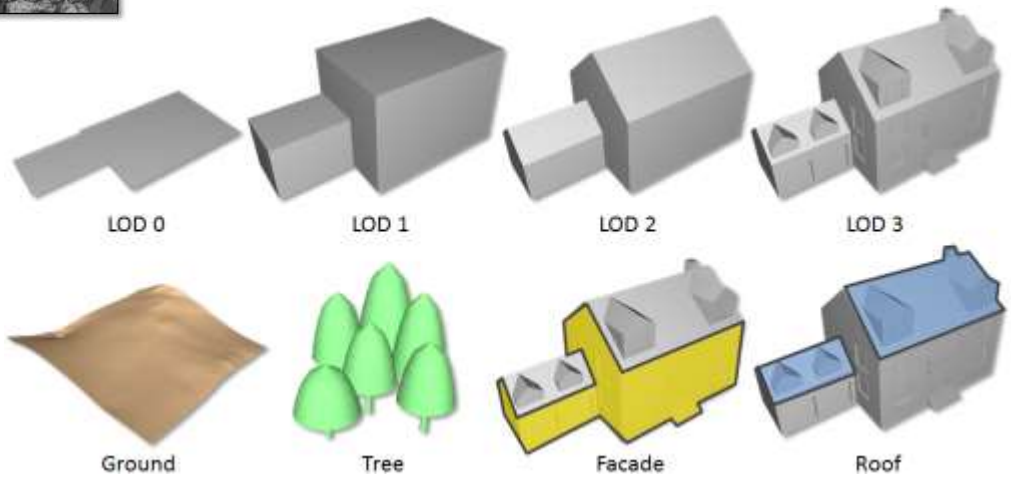
	#input points ($\times 10^6$)	area (km^2)	altimetric variation (m)	#primitives ($\times 10^3$)	#trees ($\times 10^3$)	computing time (hour)	compaction (Mo)
Marseille, France *(a)*	38.67	19.8	192	108.6	35.7	2.52	131
Amiens, France *(b)*	24.52	11.57	76	56.7	22.8	1.34	93
Mountain area *(c)*	22.67	3.41	525	0.01	21.1	0.31	34

Example 2:
Generation of urban Levels Of Detail from raw meshes

Overview

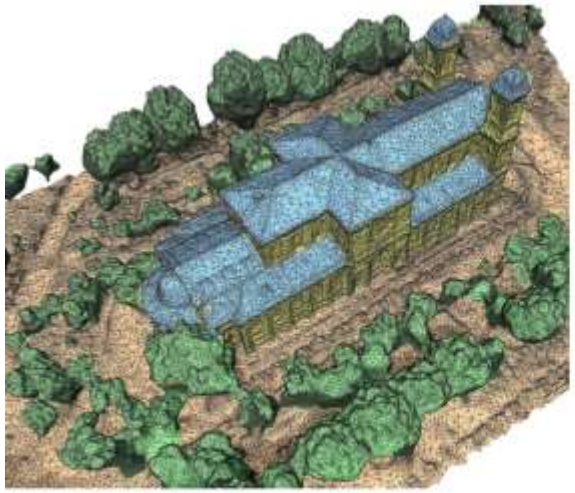


Input: raw meshes fom MVS

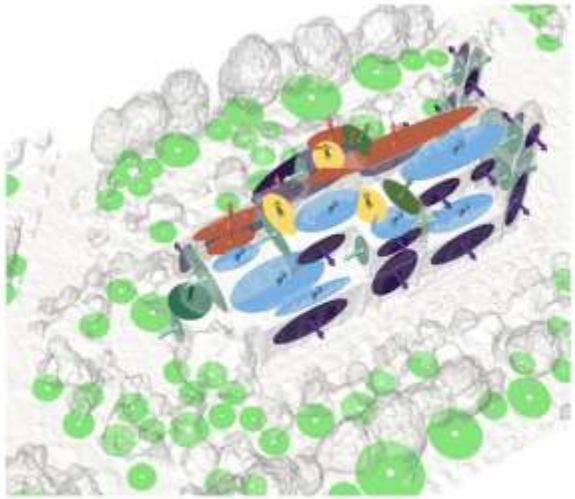


output: LOD models

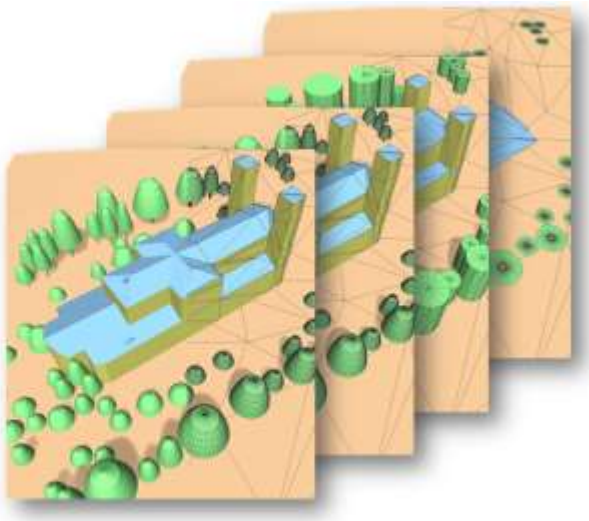
Overview



Classification



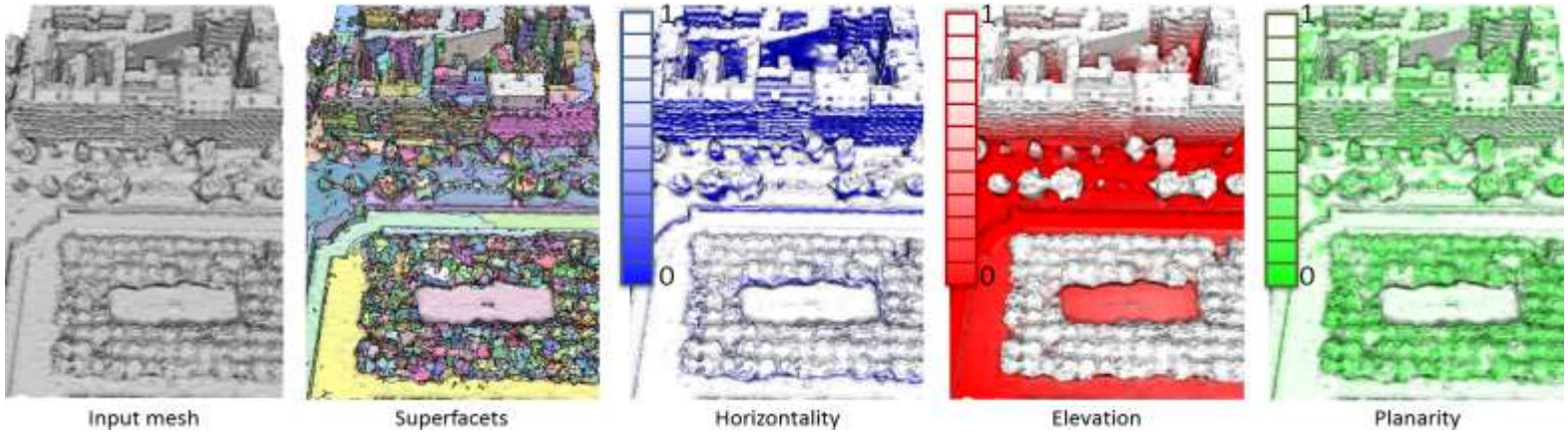
Abstraction



Reconstruction

Classification

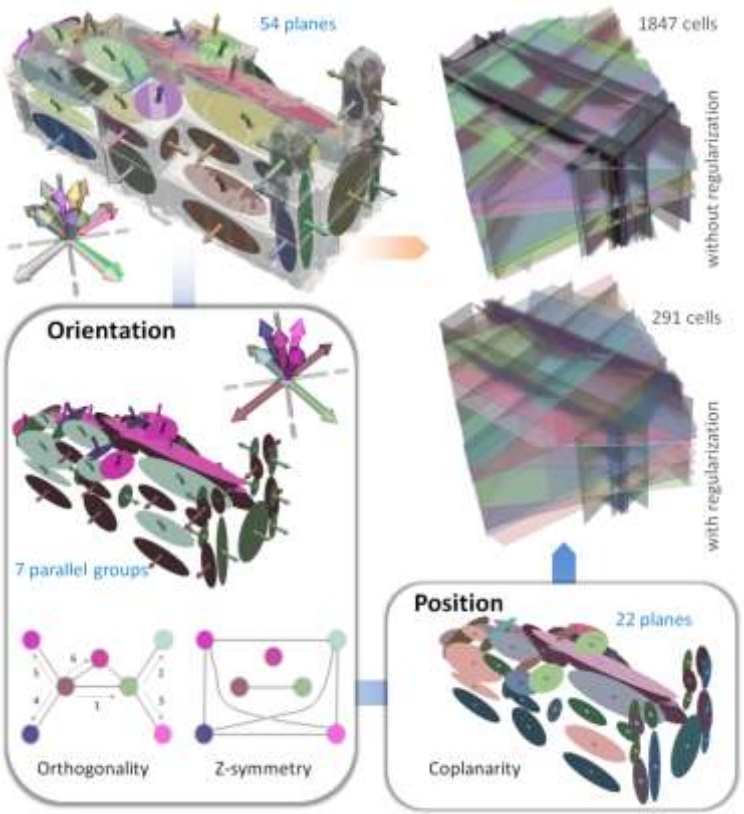
Extracting relevant geometric attributes



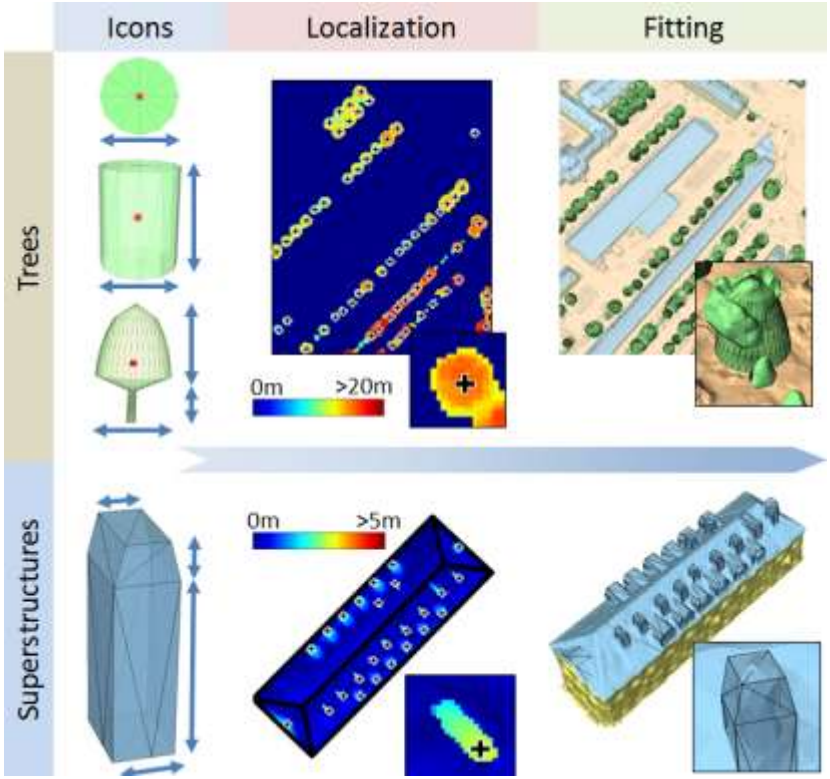
Labeling facets by MRF as
roof, facade, ground or trees

abstraction

Roofs and facades: plane detection

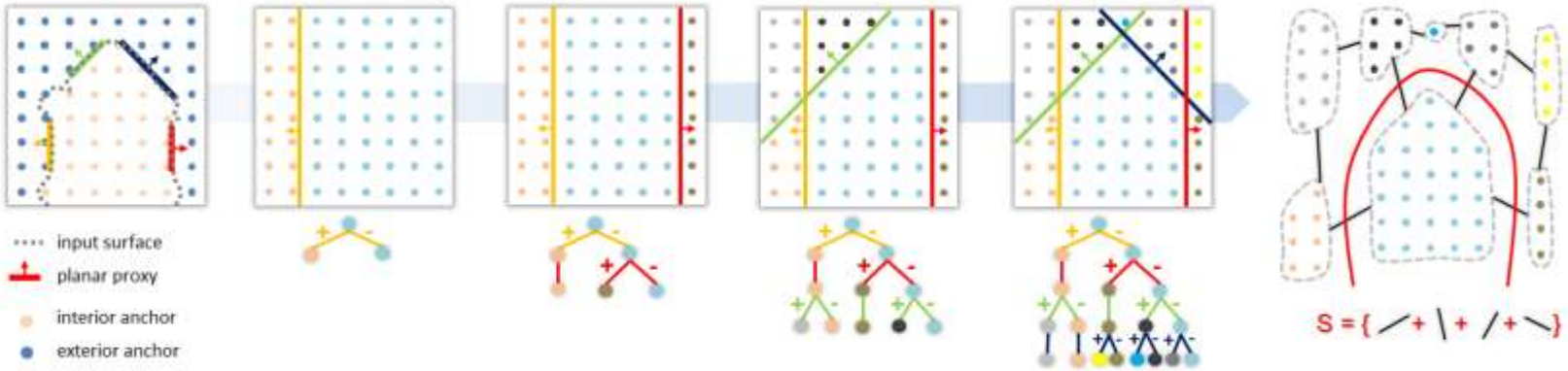


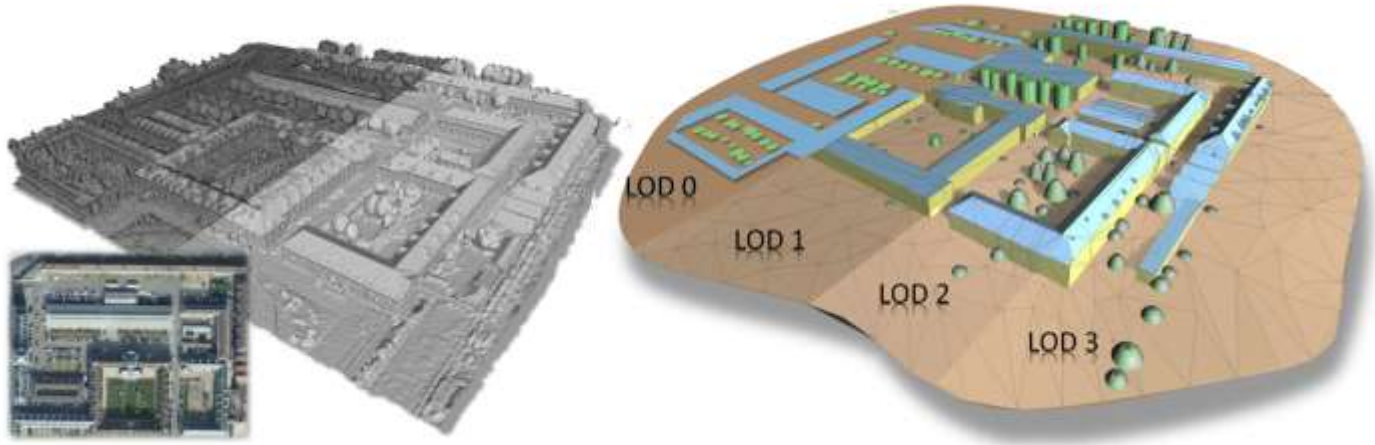
Trees and superstructures: iconization



LOD reconstruction

Space partitioning guided by planes, and inside/outside labeling of cells





Results