Institut für Kartographie und Geoinformatik | Leibniz Universität Hannover

Segmentation III

Claus.Brenner@ikg.uni-hannover.de

Leibniz Universität Hannover

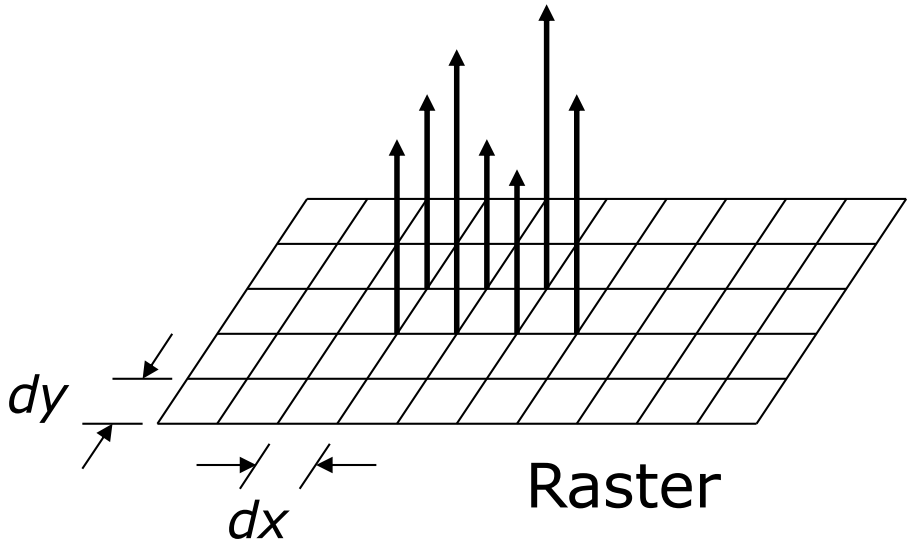
Segmentation of distance (depth) images

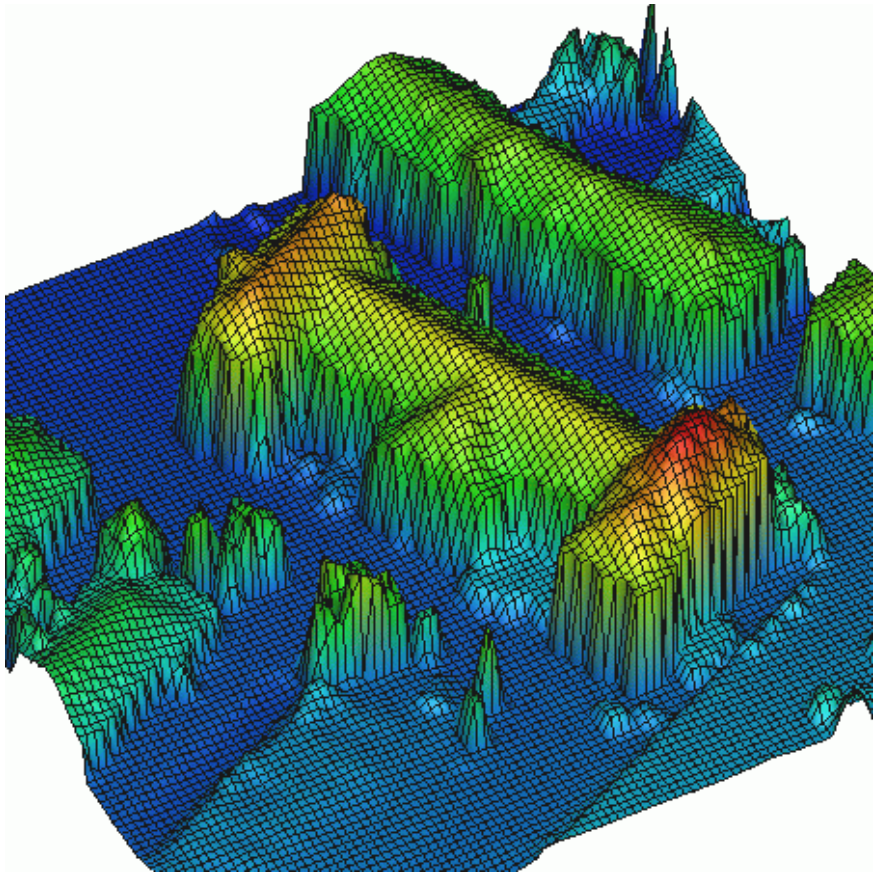
Distance images

- Segmentation of point clouds is expensive since there are no relationships between points (in the first place)
  - C.f. image: left/right/up/down neighbor (4-neighborhood), diagonal neighbors
- Often, such relationships exist, at least approximately.

Distance images

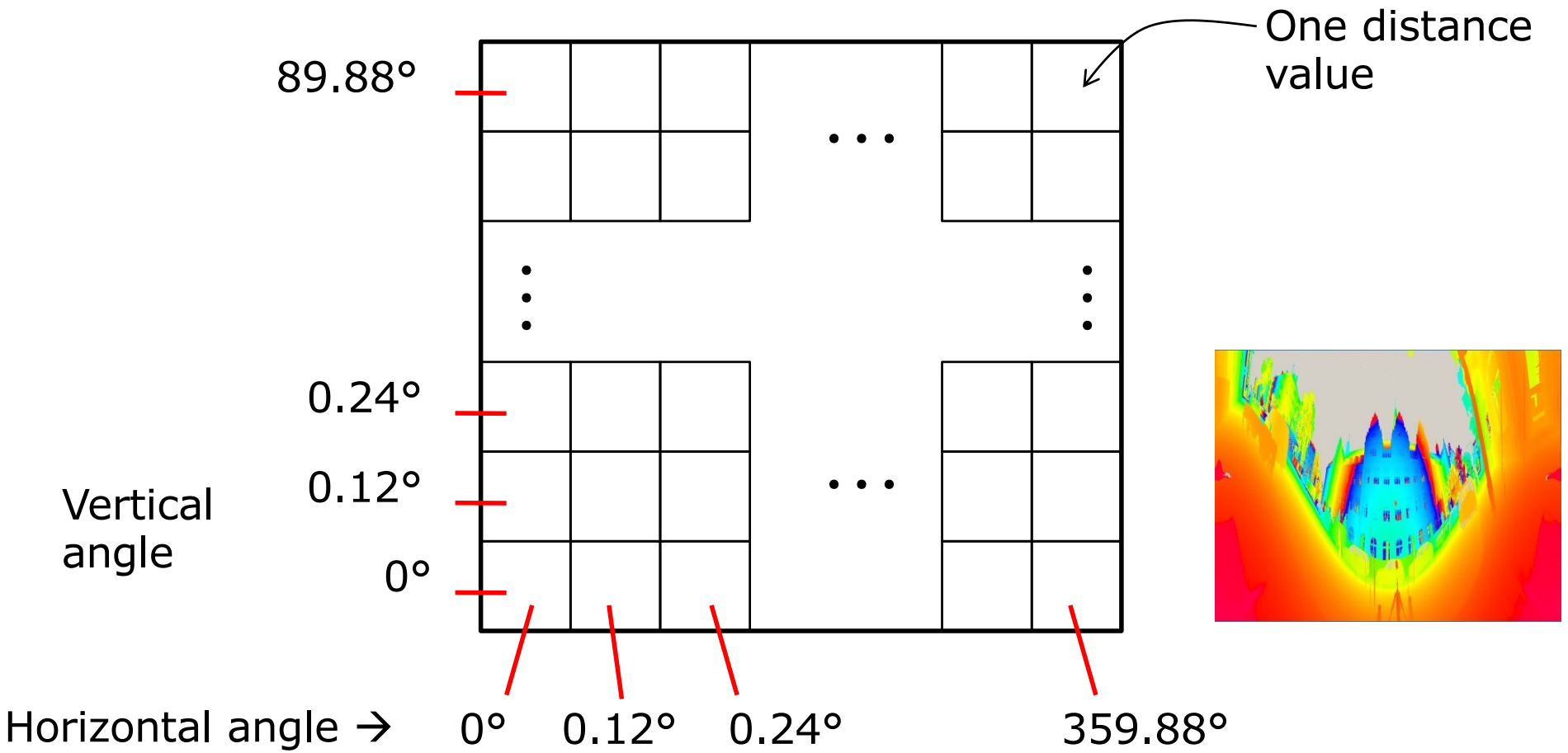
- Example 1: Height models (interpolated)





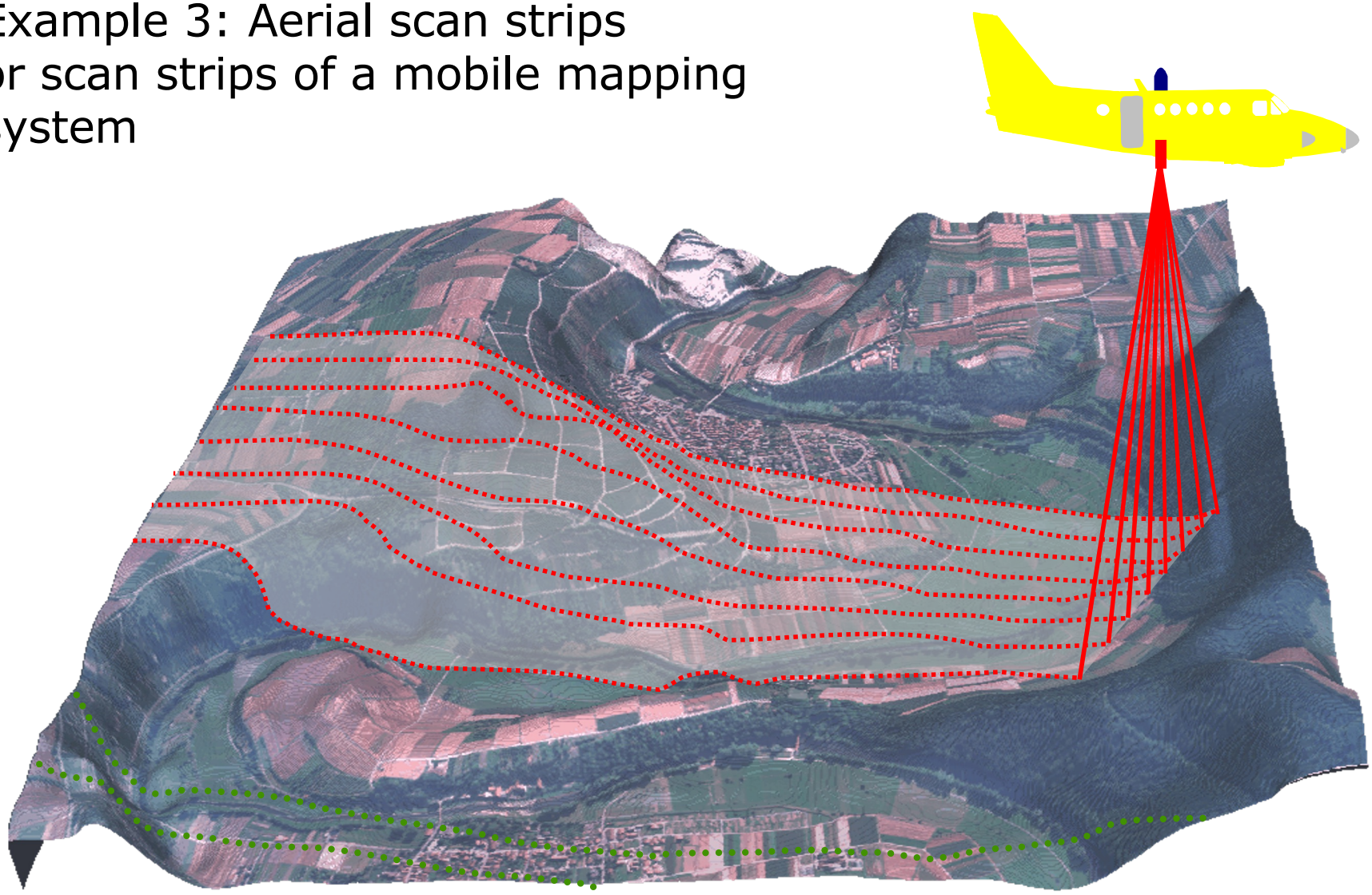
Distance images

- Example 2: terrestrial scanners – scan image (angles)

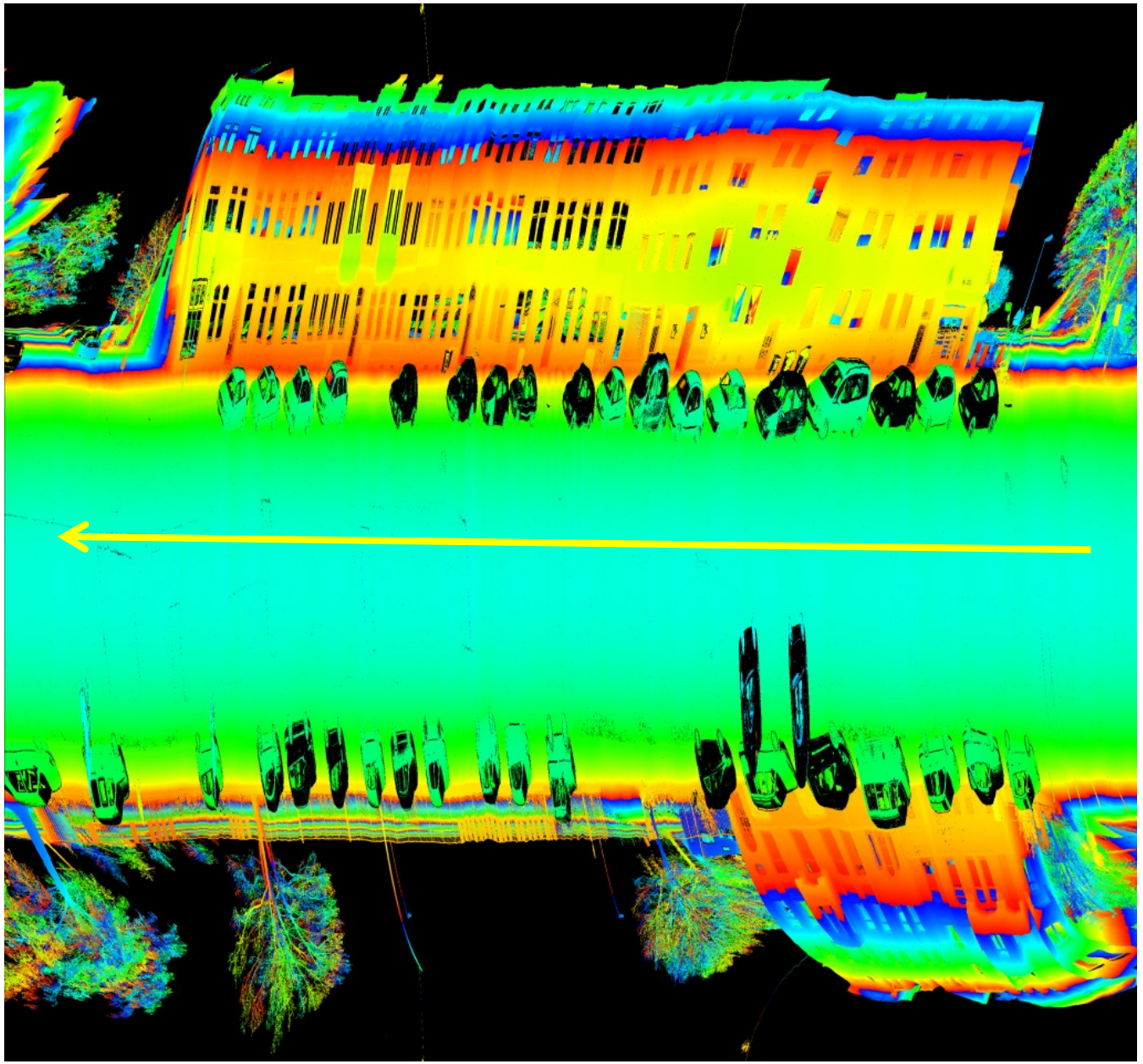


Distance images

- Example 3: Aerial scan strips or scan strips of a mobile mapping system



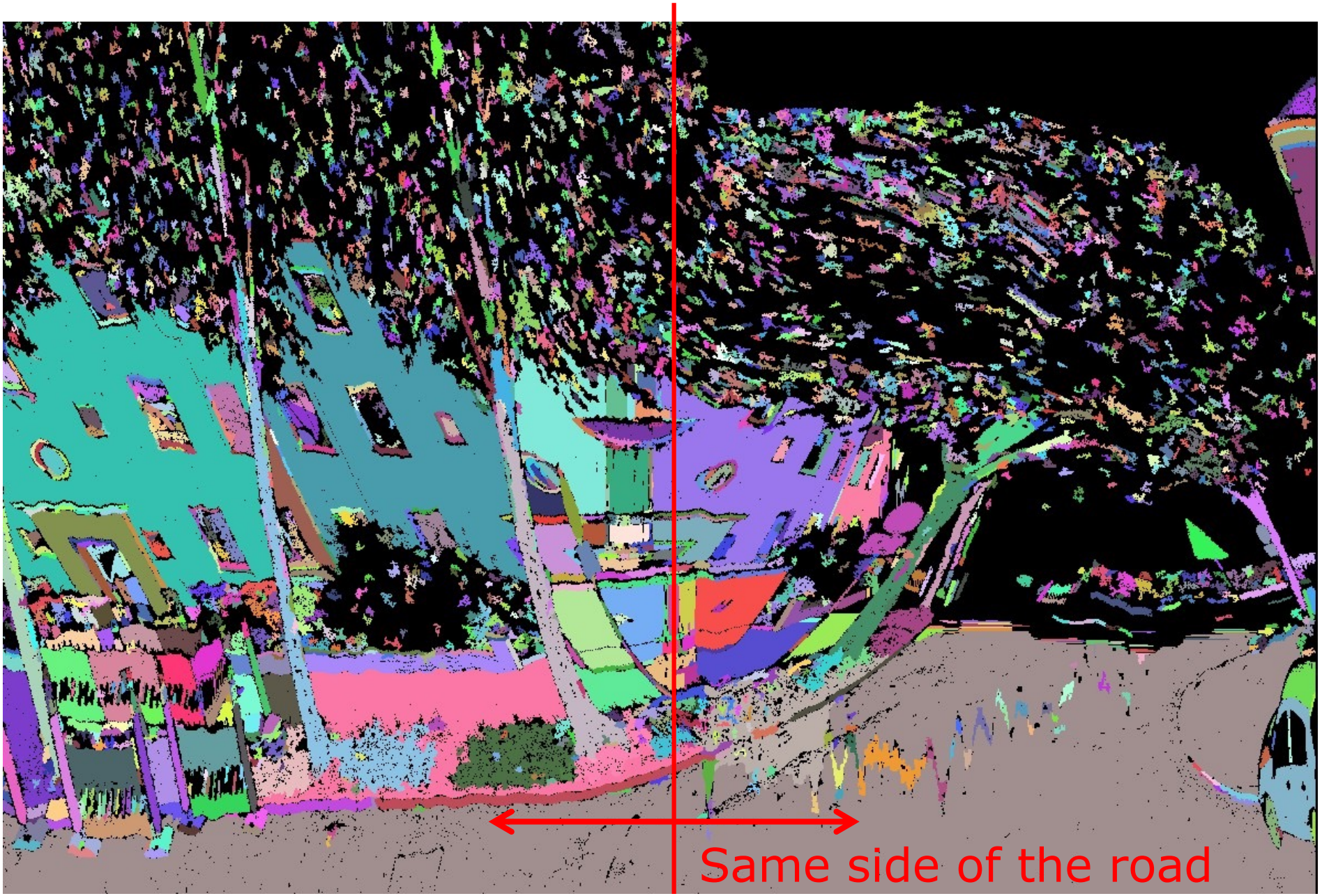
Mobile mapping scan strips (Schneiderberg)



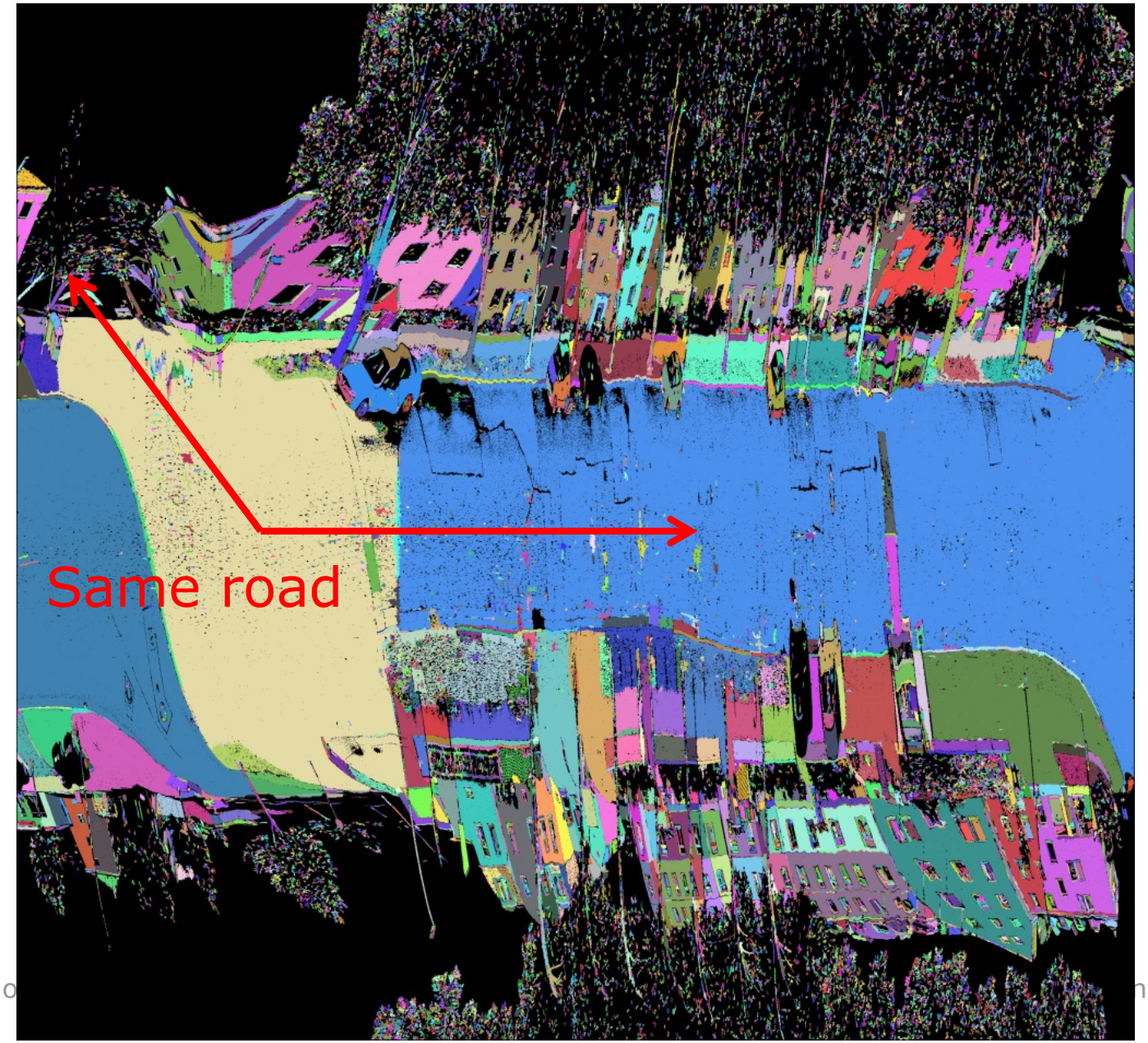
Distance images

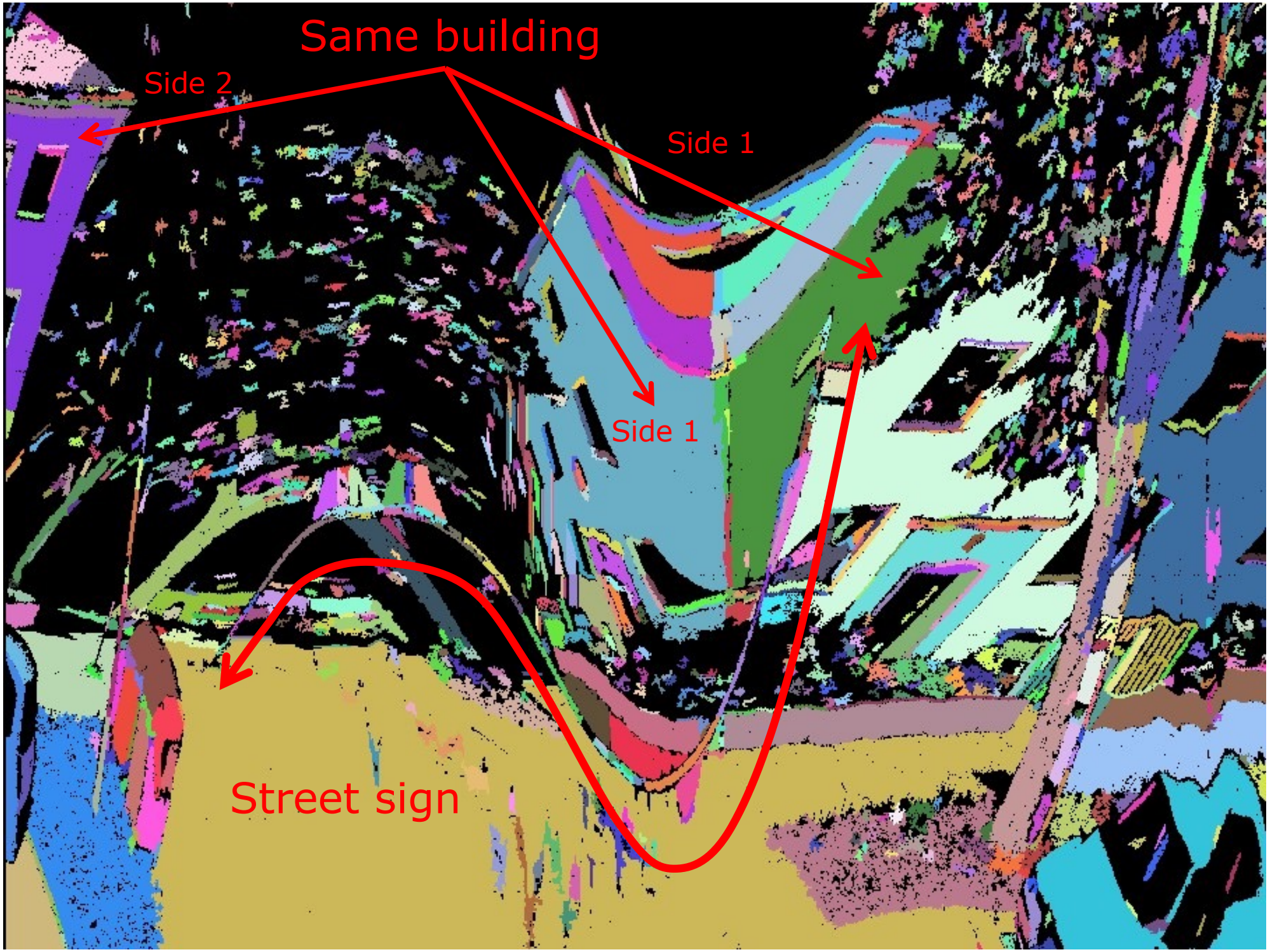
- Example 1, height model:
  - Distance image which can be written in the form $z = f(x, y)$
  - Neighbors in the xy-plane
- Example 2, terrestrial scanner:
  - Distance image which can be written in the form
    distance = f(horizontal angle, vertical angle)
    (polar coordinates)
  - Neighbors are defined in terms of the horizontal and vertical angles, not in terms of vicinity in object space
- Example 3, scan strips:
  - Rays are dependent on the orientation of the mapping platform
  - Often, subsequent measurements are close to each other in object space as well
  - But note the examples on the following slides…

Mobile mapping scan strip example (segmented)



Mobile mapping scan strip example (segmented)

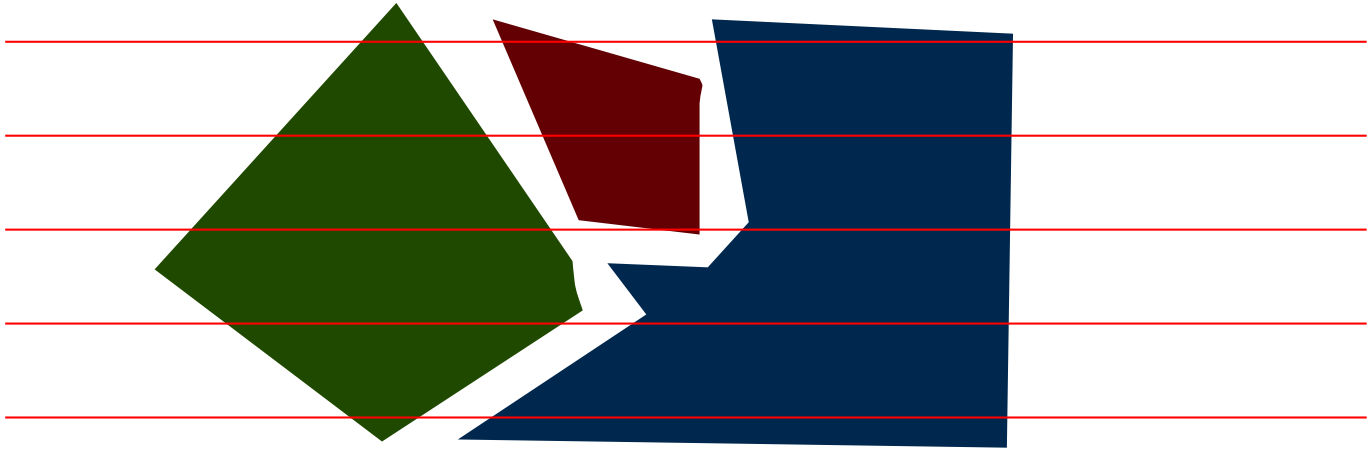



Same building
Side 2
Side 1
Side 1
Street sign

Segmentation: Scanline Grouping
Jiang & Bunke, 1994

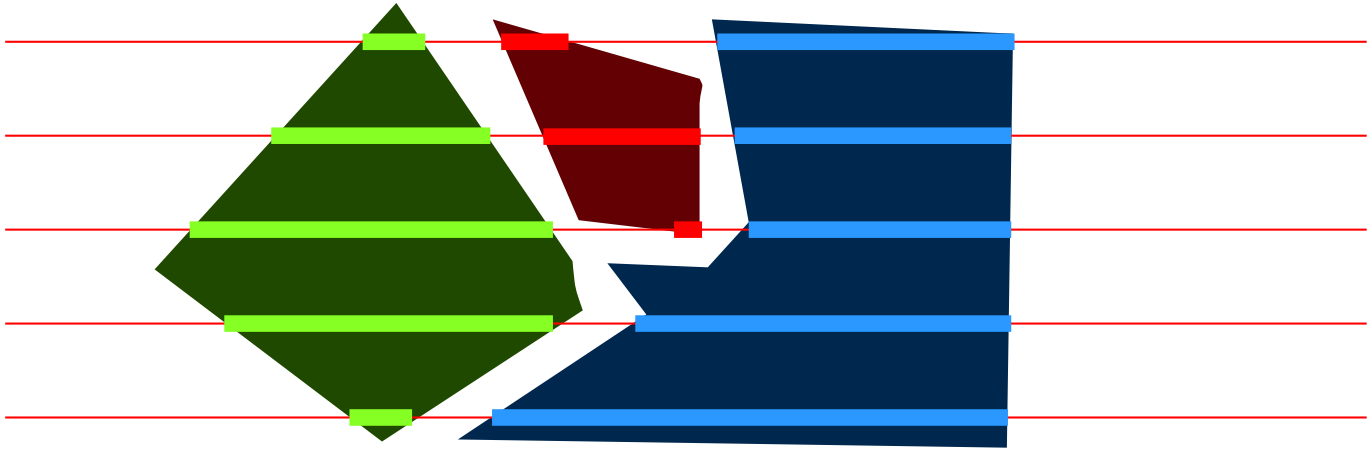
Basic idea

- Intersect scene using “scanlines” (parallel planes)
- If planes (in the scene) are intersected, this results in straight line segments



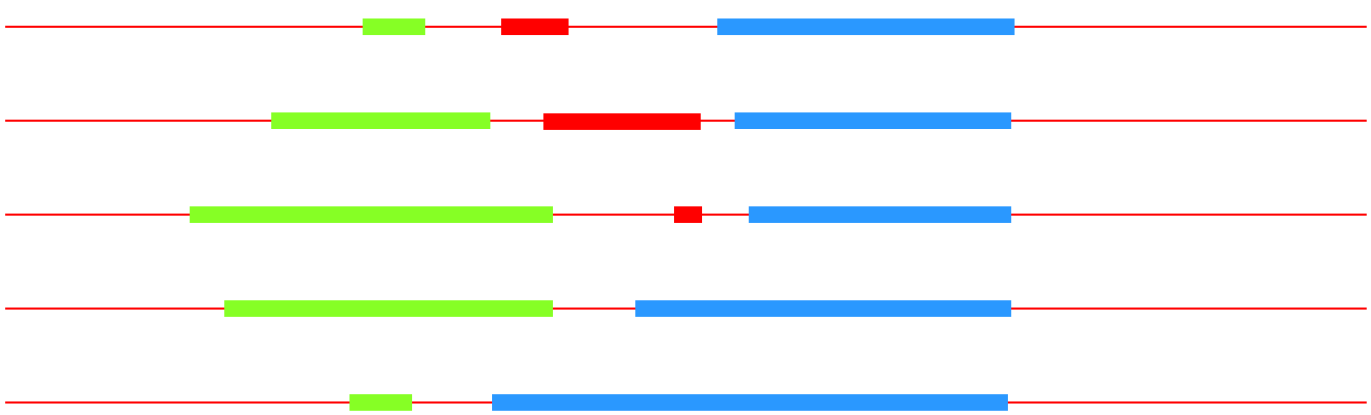
Basic idea

- Intersect scene using “scanlines” (parallel planes)
- If planes (in the scene) are intersected, this results in straight line segments



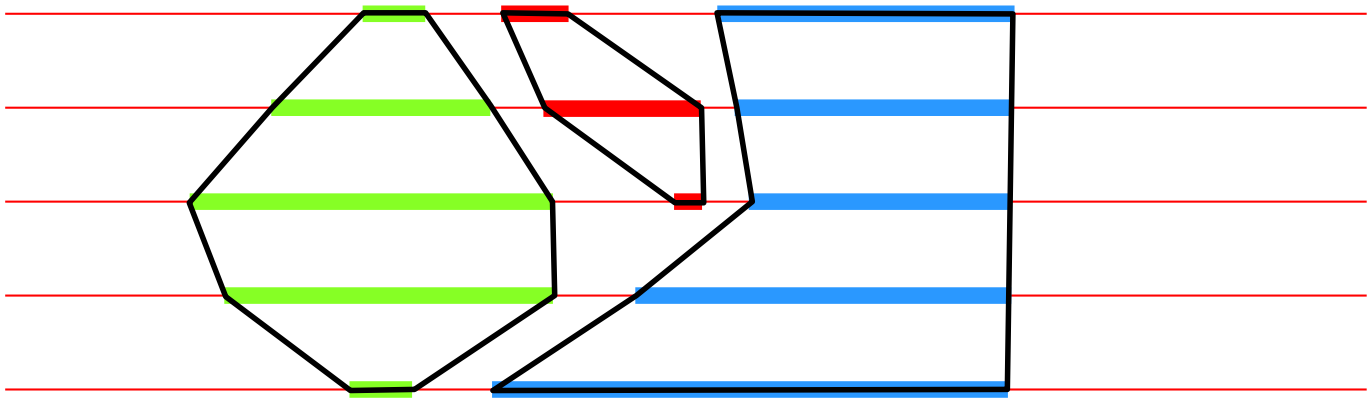
Basic idea

- Intersect scene using “scanlines” (parallel planes)
- If planes (in the scene) are intersected, this results in straight line segments
- Use this property to subdivide the scene into straight line segments



Basic idea

- Intersect scene using “scanlines” (parallel planes)
- If planes (in the scene) are intersected, this results in straight line segments
- Use this property to subdivide the scene into straight line segments
- Then, group straight line segments to planar segments

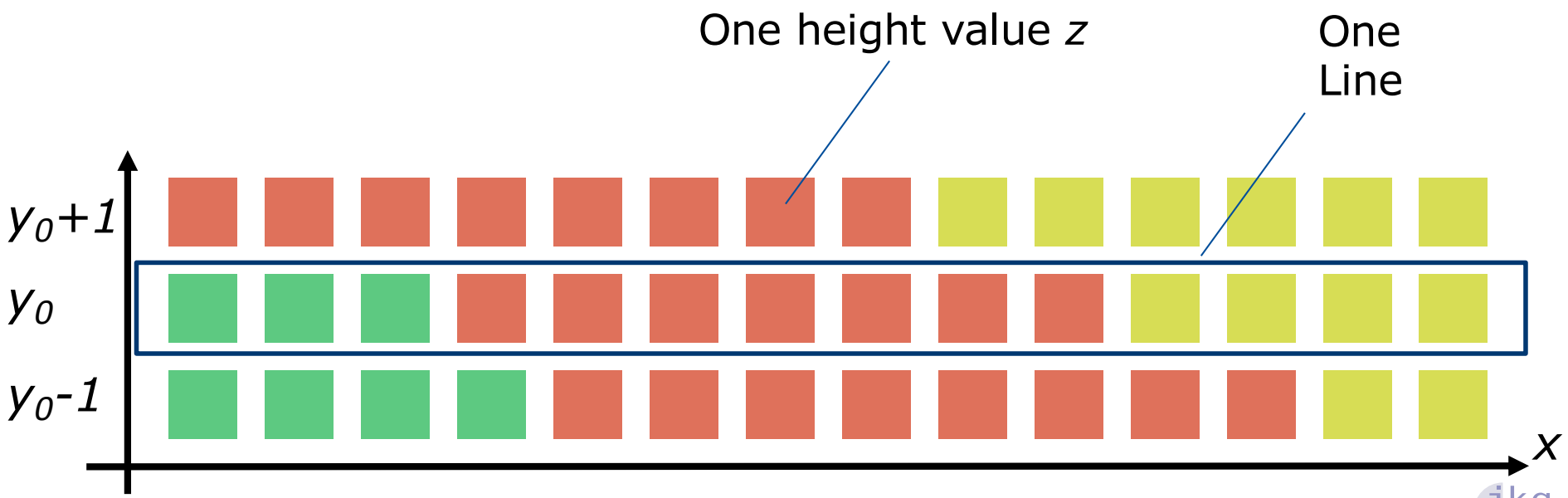


Algorithm

- Partition each row into straight line segments
- Iterative region growing:
  - Select best seed region (consisting of 3 segments in successive rows)
  - Grow the region by adding more segments
  - If further growth is not possible, accept region and select next (best) seed region
- Use post-processing to smooth region borders

Scanline grouping – Step 1

- Partition each scanline into straight line segments



Scanline grouping – Step 1

- Assume the "red plane" fulfills the plane equation

$$z = ax + by + c$$

- Then, along the scan line $y = y_0$ the points fulfill:

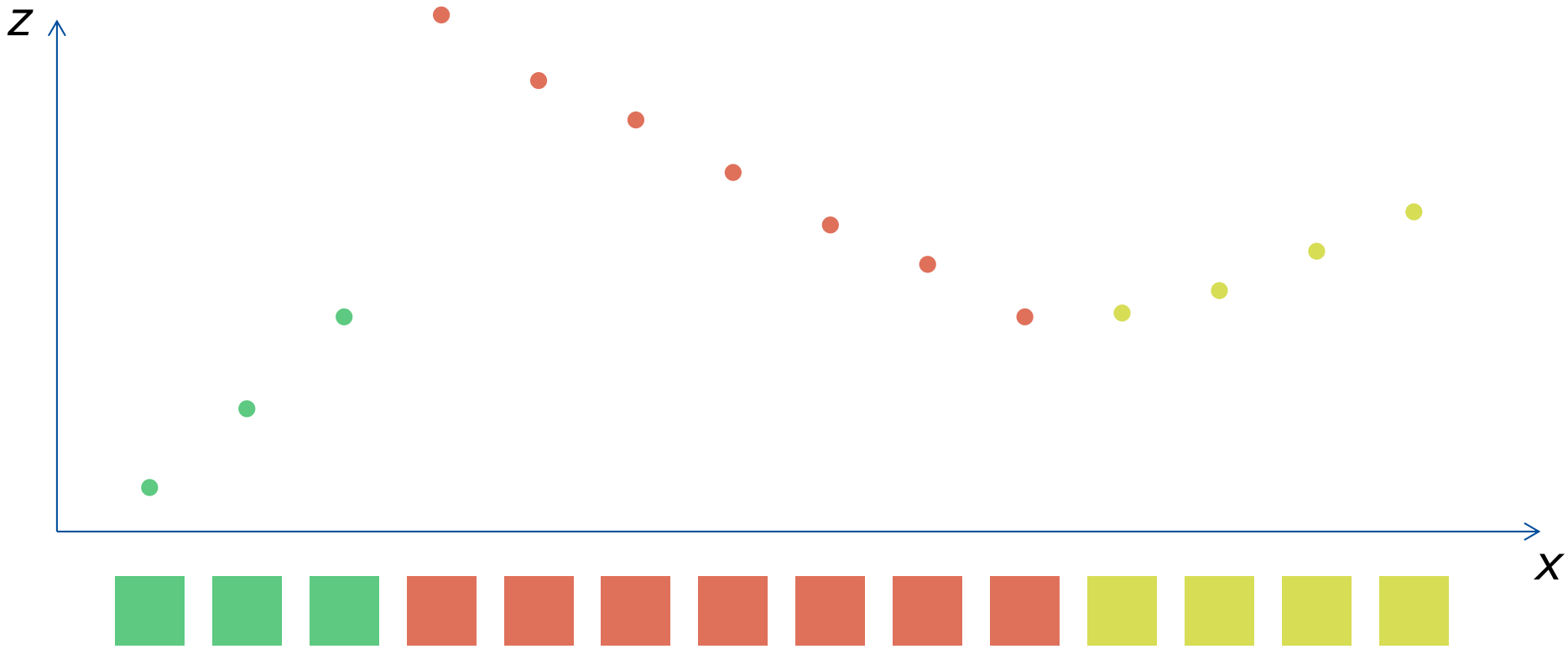
$$z = ax + \underbrace{by_0 + c}_{=:b_0}$$

- This is a line equation:

$$z = ax + b_0$$

Scanline grouping – Step 1

- Example profile along scan line $y = y_0$:



Scanline grouping – Step 1

- Segment scan line, e.g. by using algorithm by Douglas-Peucker

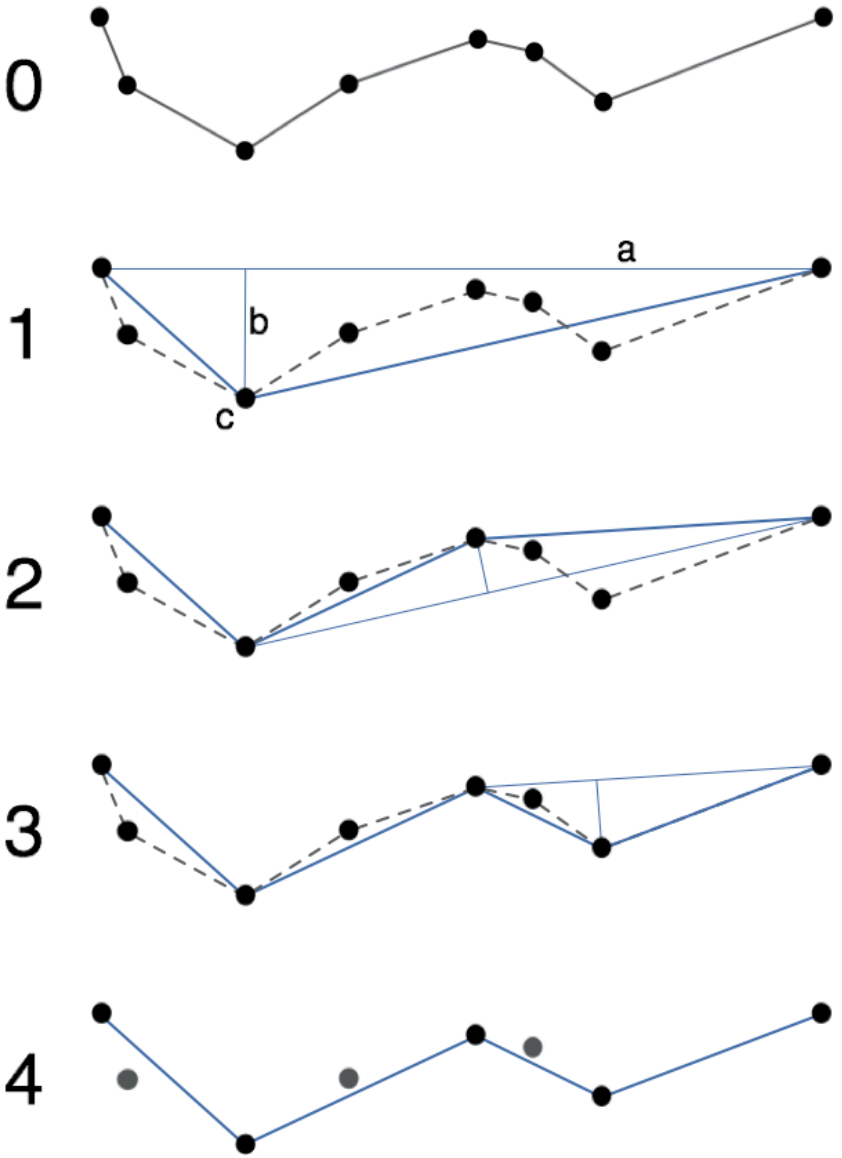
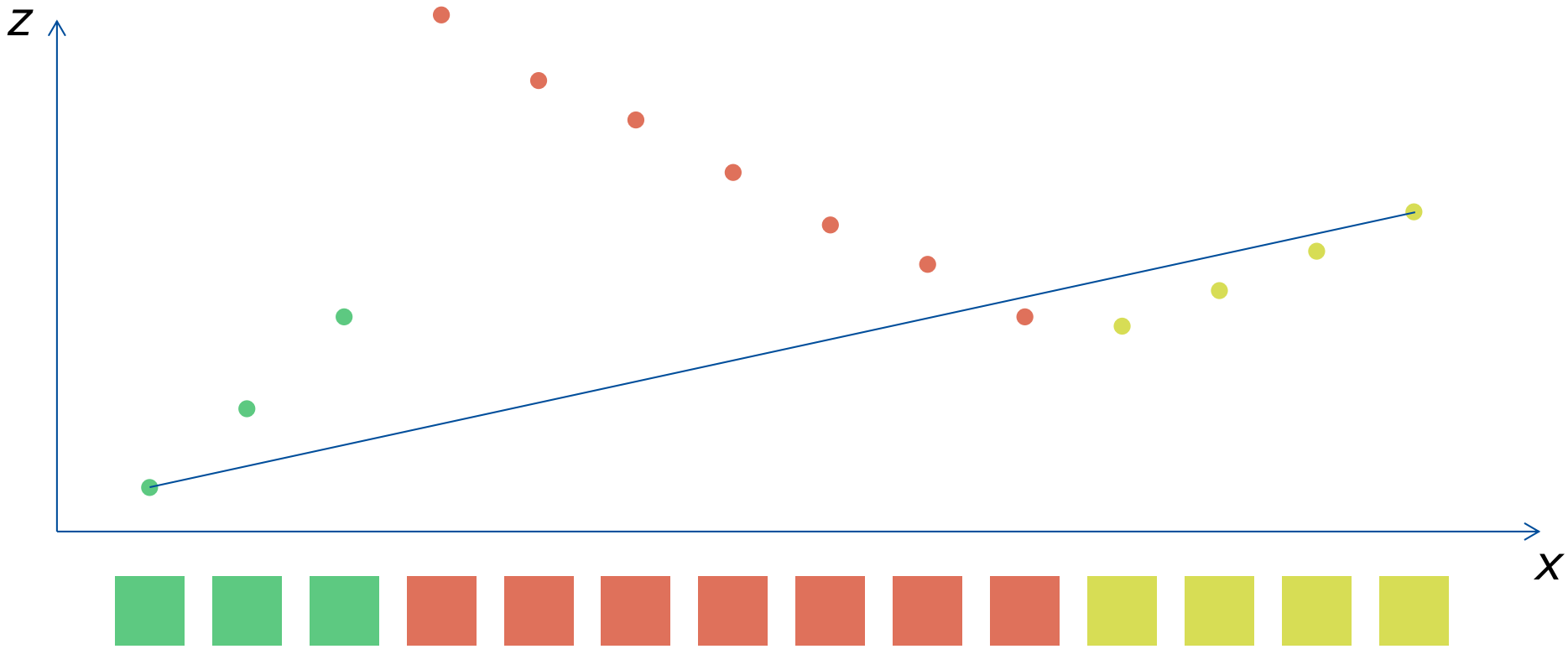


Image source: Wikipedia

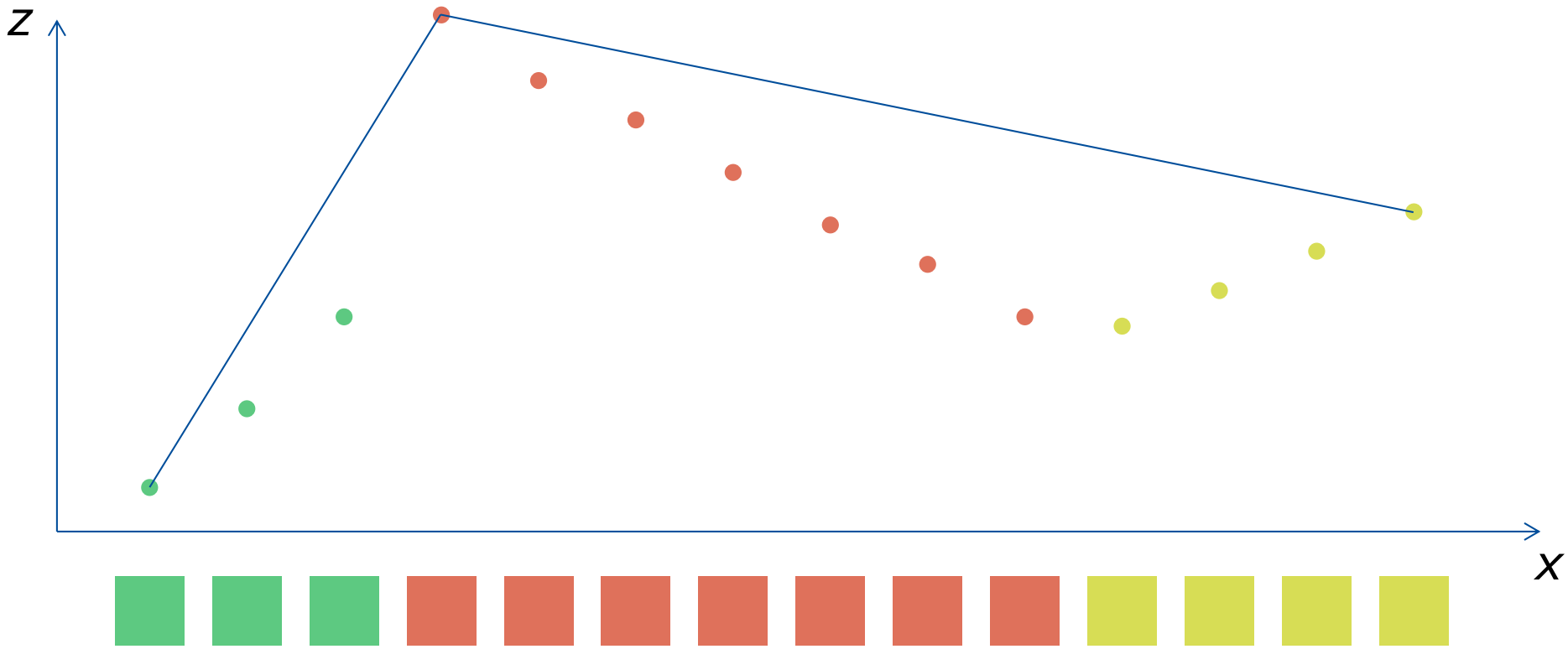
Scanline grouping – Step 1

- Segmentation using Douglas-Peucker algorithm



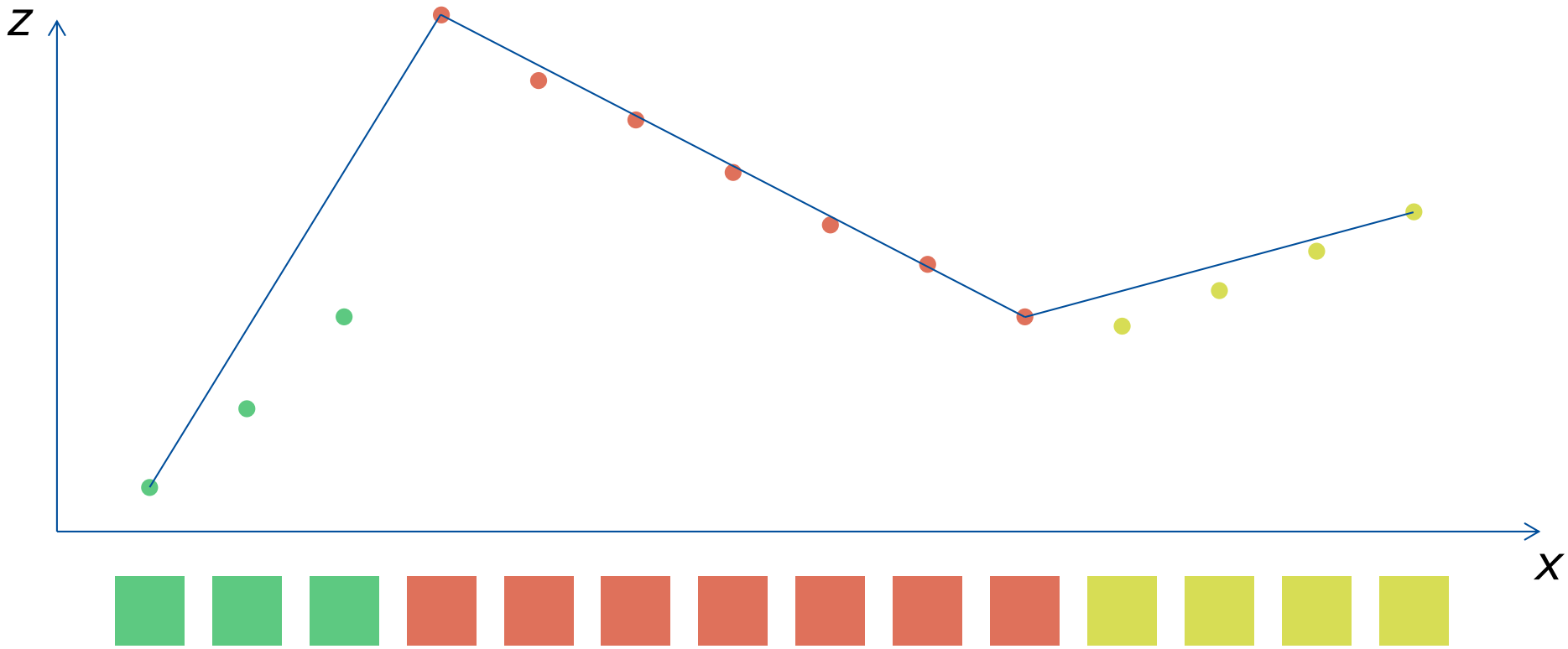
Scanline grouping – Step 1

- Segmentation using Douglas-Peucker algorithm



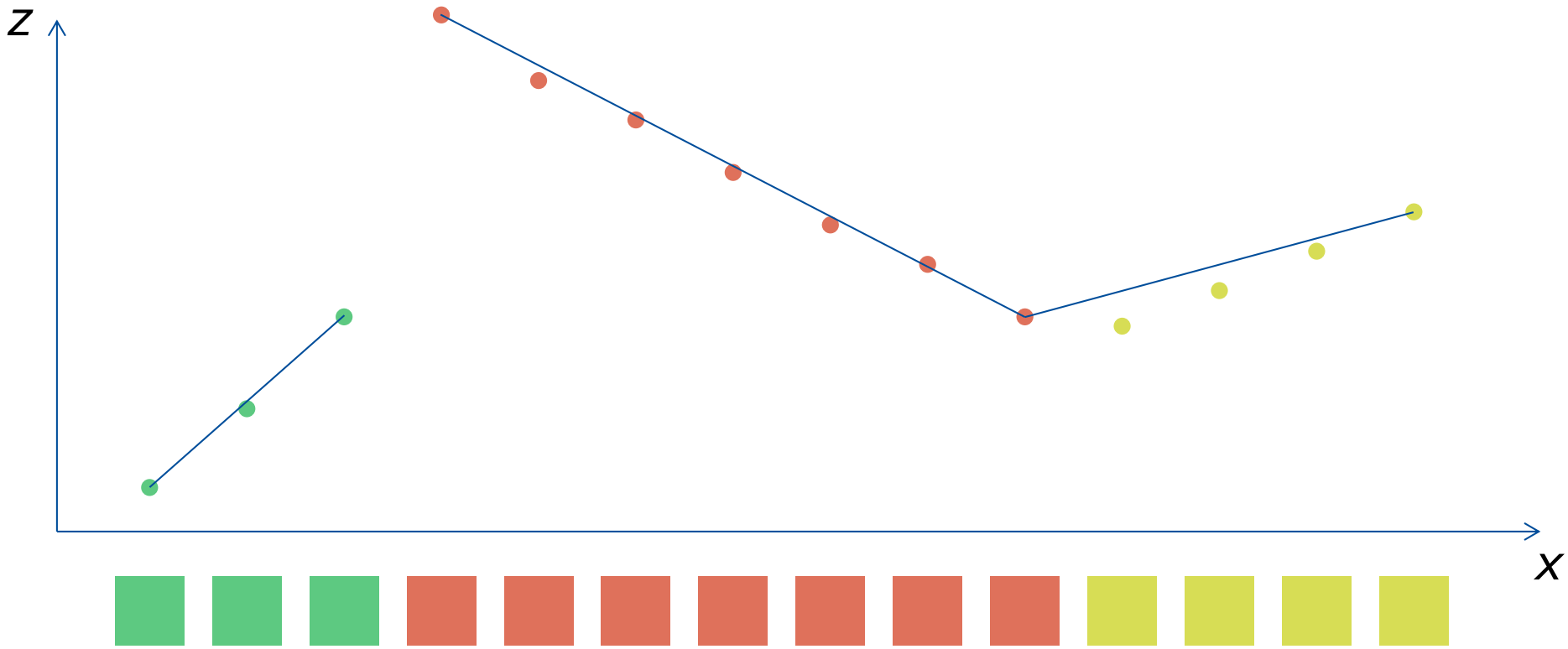
Scanline grouping – Step 1

- Segmentation using Douglas-Peucker algorithm



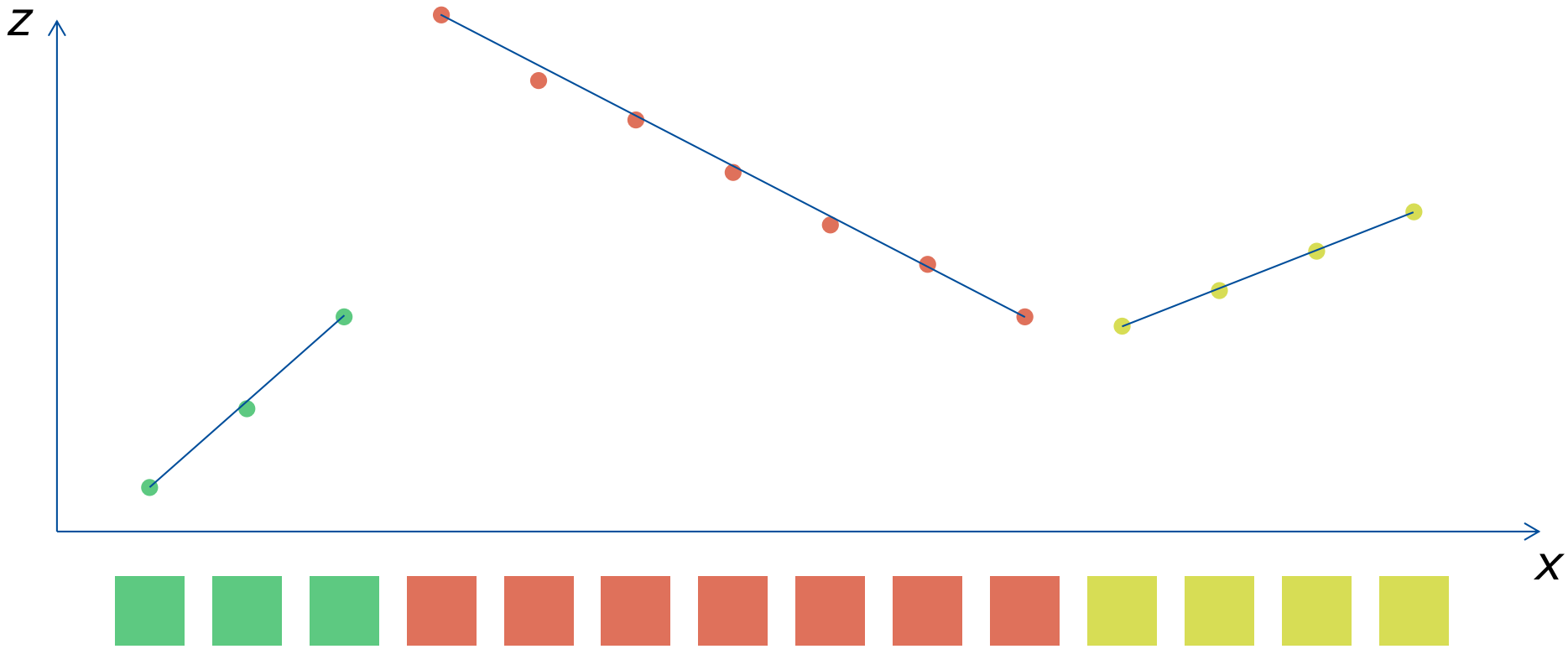
Scanline grouping – Step 1

- Segmentation using Douglas-Peucker algorithm



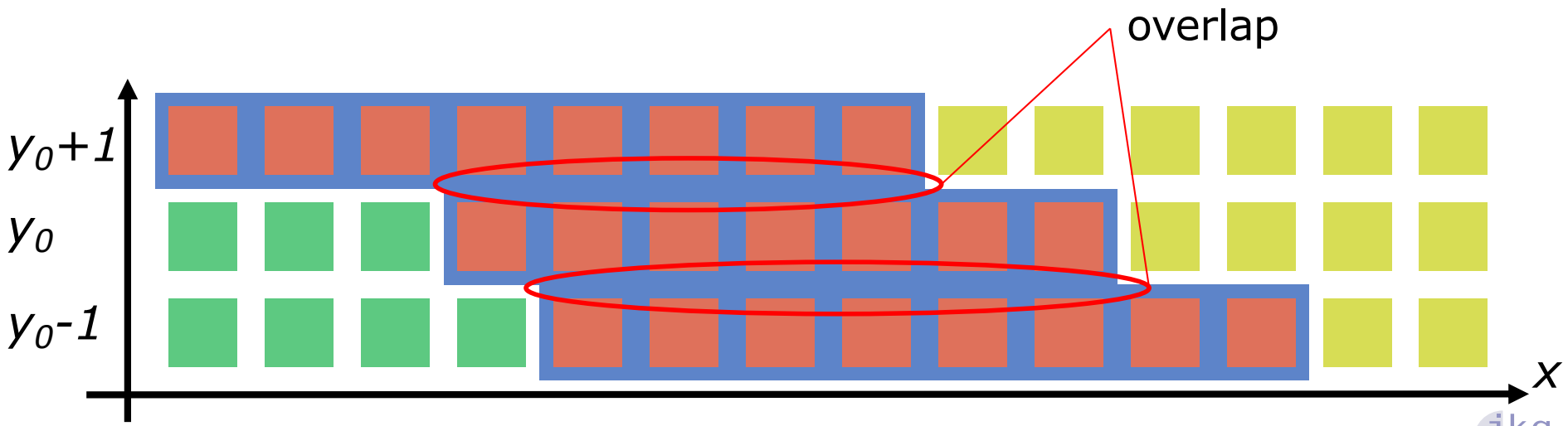
Scanline grouping – Step 1

- Segmentation using Douglas-Peucker algorithm



Scanline grouping – Step 2a

- Region growing
  - Find seed regions
  - Use segments which overlap in 3 successive rows
  - Compute a “quality” for each seed region



Scanline grouping – Step 2a

- Quality of the seed region
  - One idea: deviation of points from a mean plane
  - Possible implementation:
    - Compute mean (least squares) plane through all points of the three segments
    - Use standard deviation (point residuals) as “quality” measure
  - Drawback: each point has to be “touched” in order to compute the plane. This would negate the advantage of scanline grouping
- Instead, Jiang & Bunke use a modified criterion, which can be computed from the segments directly
  - I.e. without touching individual points

Scanline grouping – Step 2a

- Assume: the three selected segments are indeed part of this plane:

$$z = ax + by + c$$

- Then, this holds for the three segments:

$$s_{-1}: \quad z = ax + b(y_0 - 1) + c = ax + b_0 - b$$

$$s_0: \quad z = ax + by_0 + c = ax + b_0$$

$$s_{+1}: \quad z = ax + b(y_0 + 1) + c = ax + b_0 + b$$

- They all share the same slope, *a*
- The z-intercept of successive segments differs by *b*

Scanline grouping – Step 2a

- Assume the segment equations of the 3 (“real”) segments are:

$$s_{-1}: \quad z = a_{-1}x + b_{-1}$$
$$s_{0}: \quad z = a_{0}x + b_{0}$$
$$s_{1}: \quad z = a_{1}x + b_{1}$$

- Now, define the normal vector (for each segment)

$$\mathbf{m}_i = \begin{bmatrix} a_i \\ -1 \end{bmatrix}$$

- Then, the similarity of the slope of two segments can be defined in terms of the scalar product (cosine of the angle between the normal vectors):

$$\text{similarity\_a} = \frac{\langle \mathbf{m}_i, \mathbf{m}_j \rangle}{\|\mathbf{m}_i\| \|\mathbf{m}_j\|}$$

Scanline grouping – Step 2a

- Similar approach for the z-intercept
- The difference between intercepts should be (approximately) *b*
- Define "pseudo normal vectors":

$$\mathbf{n}_{-1} = \begin{bmatrix} b_0 - b_{-1} \\ -1 \end{bmatrix} \qquad \mathbf{n}_0 = \begin{bmatrix} b_1 - b_0 \\ -1 \end{bmatrix} \qquad \mathbf{n}_1 = \begin{bmatrix} (b_1 - b_{-1})/2 \\ -1 \end{bmatrix}$$

- Define the similarity (analogous to slope case on previous slide):

$$\text{similarity\_b} = \frac{\langle \mathbf{n}_i, \mathbf{n}_j \rangle}{\|\mathbf{n}_i\| \|\mathbf{n}_j\|}$$

Scanline grouping – Step 2a

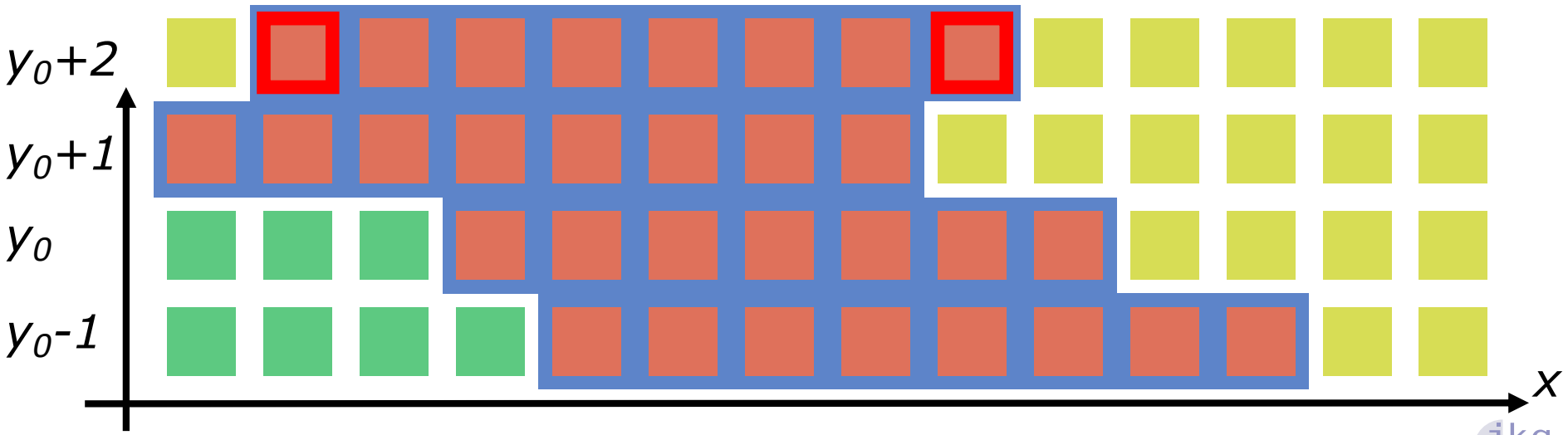
- Combine all criteria into one single "quality" estimate:

$$quality := \frac{1}{12}\left(\sum_{i<j}\frac{\langle \mathbf{m}_i, \mathbf{m}_j \rangle}{\|\mathbf{m}_i\|\|\mathbf{m}_j\|} + \sum_{i<j}\frac{\langle \mathbf{n}_i, \mathbf{n}_j \rangle}{\|\mathbf{n}_i\|\|\mathbf{n}_j\|}\right) + \frac{1}{2} \quad \in [0,1]$$

- (Between 0 and 1, where a perfect fit results in 1)
- Advantage:
  - Independent of the number of points (in the seed region), this "quality" can be computed using two sums (of 3 terms each)

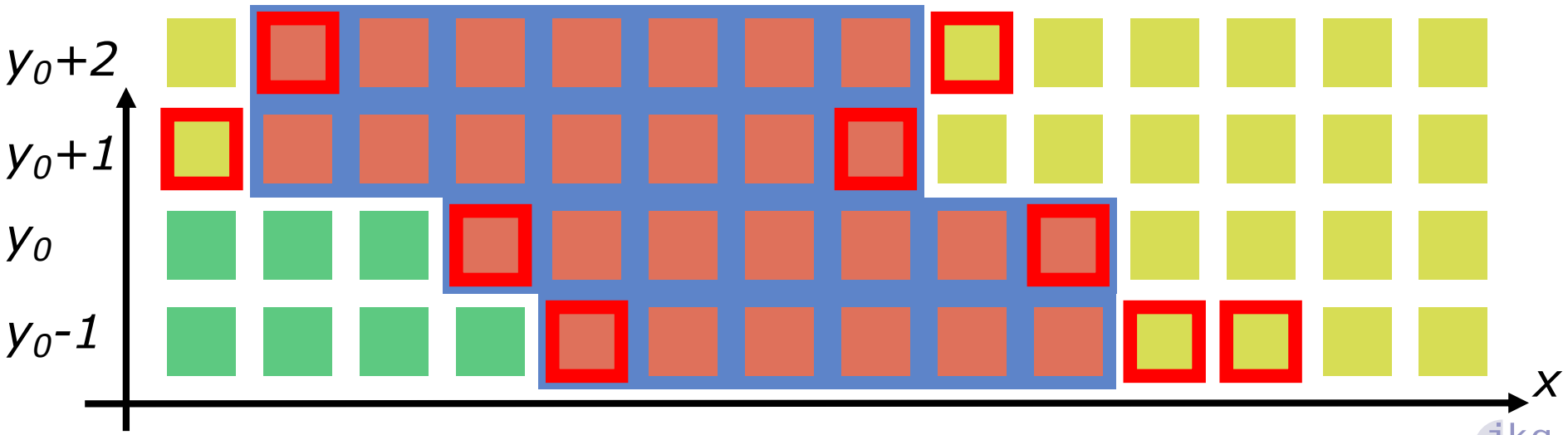
Scanline grouping – Step 2b

- Grow the region
- In each step, decide if a scanline (segment) shall be added
- To decide, only the two endpoints have to be tested, since the max. deviations can only occur there
- → Very efficient!



Scanline grouping – Step 3

- Post-processing
- Check if points at the (horizontal) region borders should be re-assigned
- Iterate until the result is stable (no re-assignments)

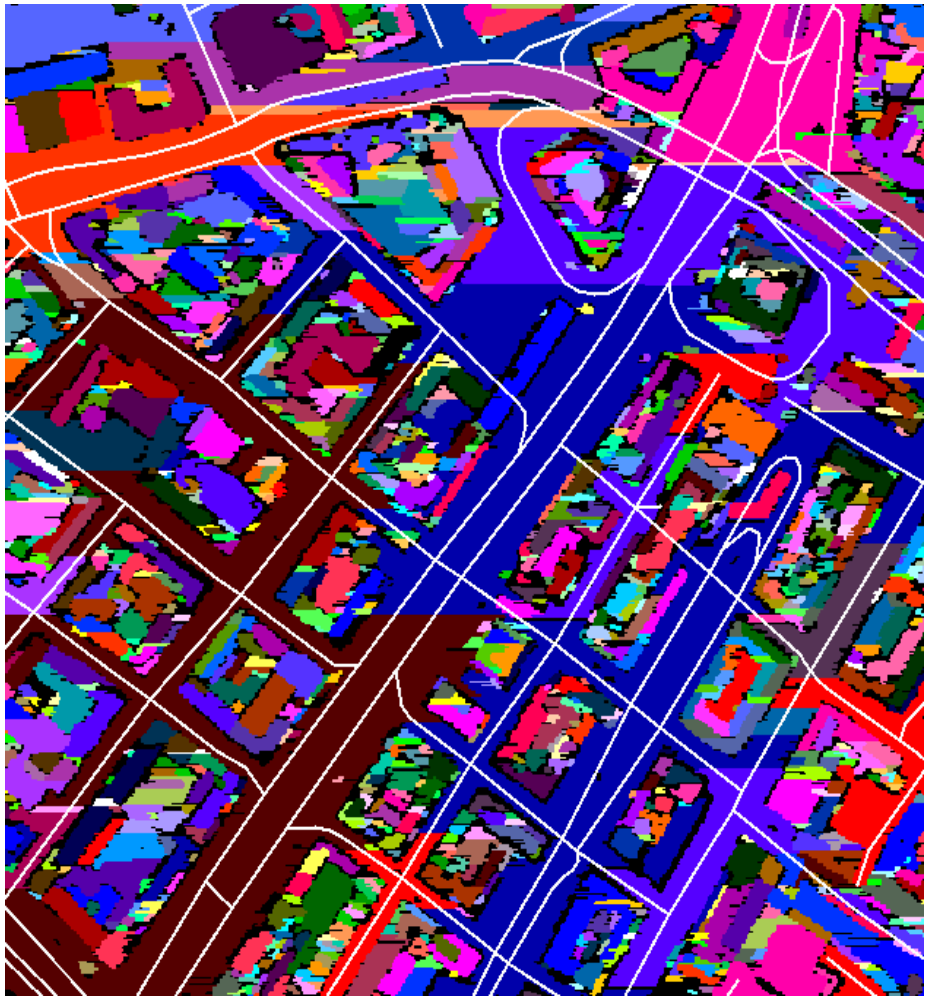


Further remarks: scanline grouping

- Similar to region growing
- But: difference during region growth:
  - Standard region growing adds elements pixel by pixel
  - Scanline grouping adds entire segments
- Therefore, scanline grouping is usually much faster than standard region growing
- Example: z = f(x, y) (Example 1, height model)
  - Could be modified to work for terrestrial scanning or aerial/ mobile mapping scan strips as well.

Scanline grouping: Example

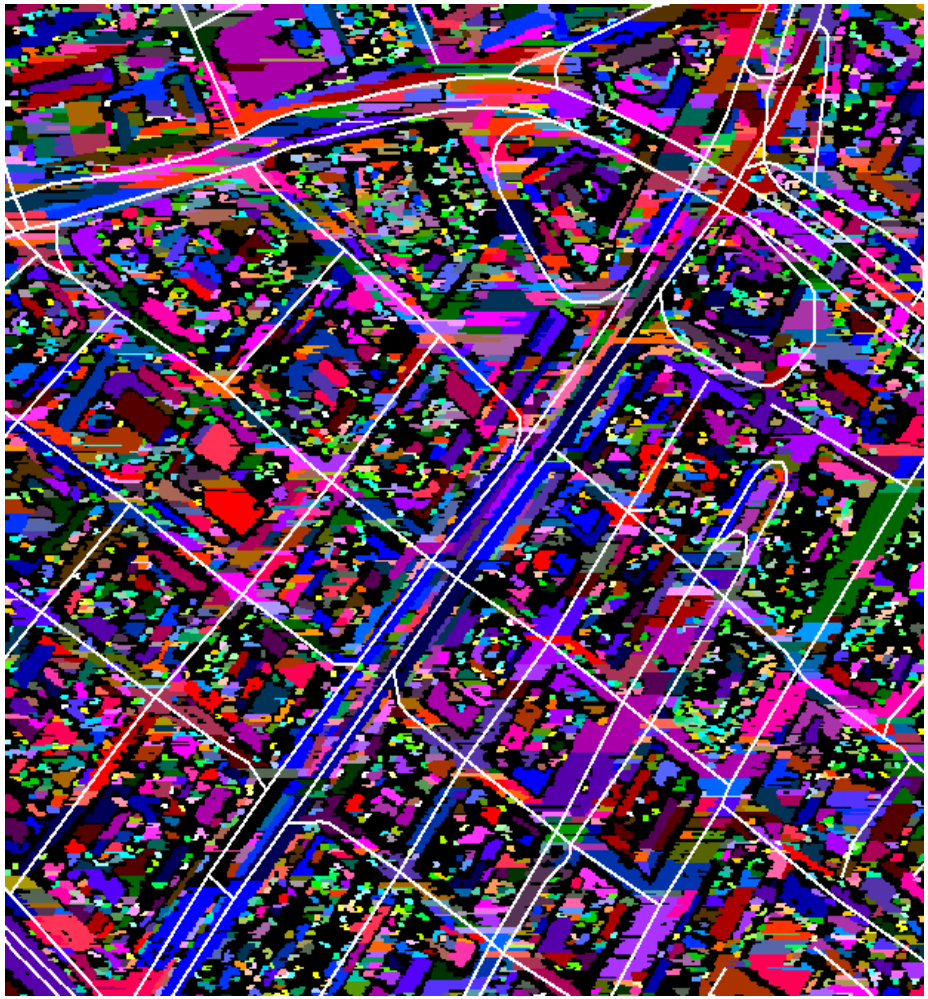
DSM (1m)



Segmentation

Scanline grouping: Example

DSM (1m)



Segmentation

Scanning and segmentation in robotics applications

Positioning

- Alignment of scan data does not only yield a merged point cloud, but also determines the position and orientation of the sensor at the time of data capture
- Therefore, the data itself can be used to recover the position and orientation of a (scanning) robot

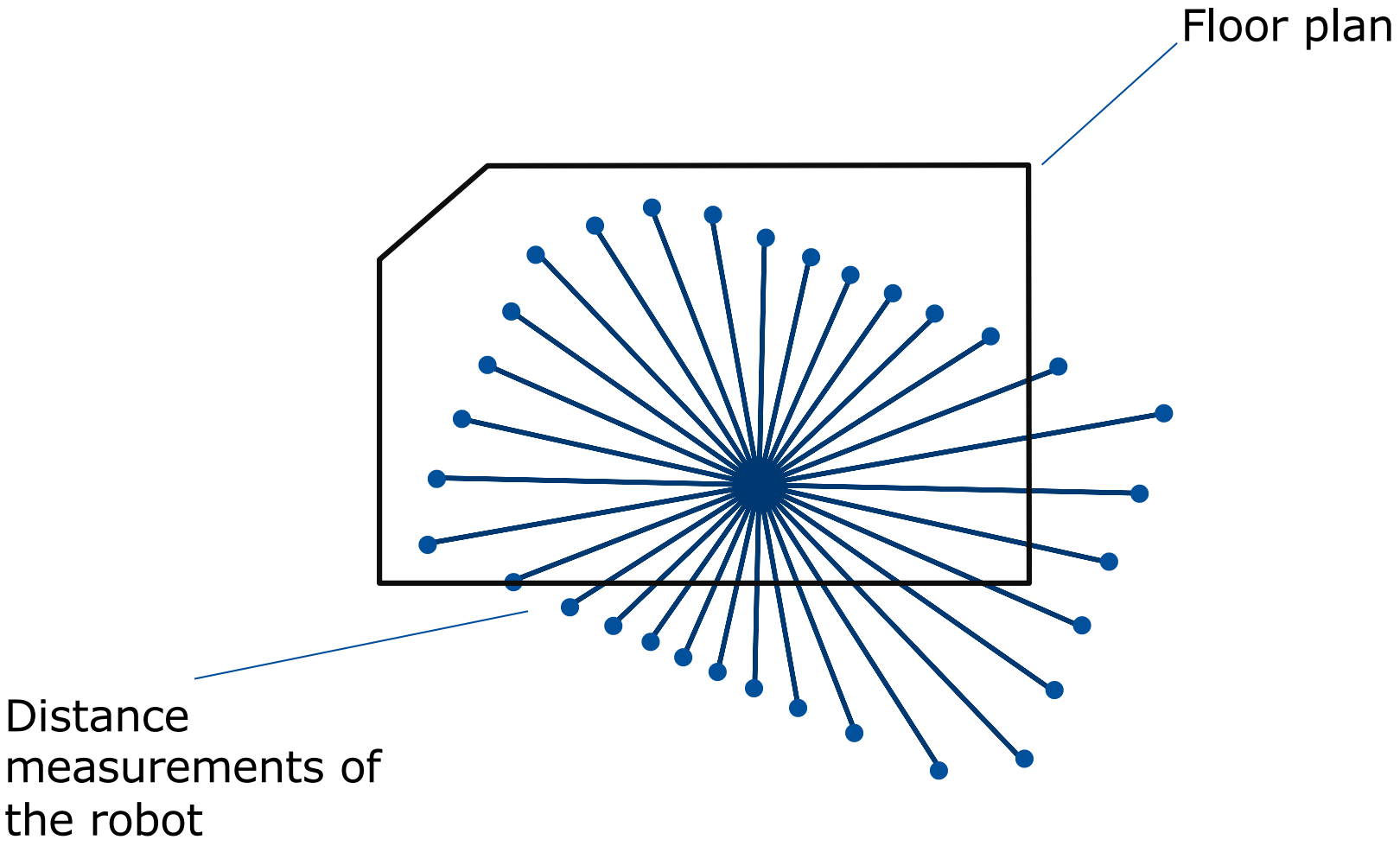
- Two (major) cases:
  - The environment **is given** in terms of a “map”. The task is to align the scan data to the map.
  - The environment **is unknown**. The robot aligns all scans successively. This is termed “simultaneous localization and mapping”, SLAM.

Fine localization

- Assume the map is given
- E.g.: ground (floor) plan
- If the approximate position of the robot is known the data can be aligned using the iterative closest point algorithm
- This also determines the position of the robot
- = fine localization

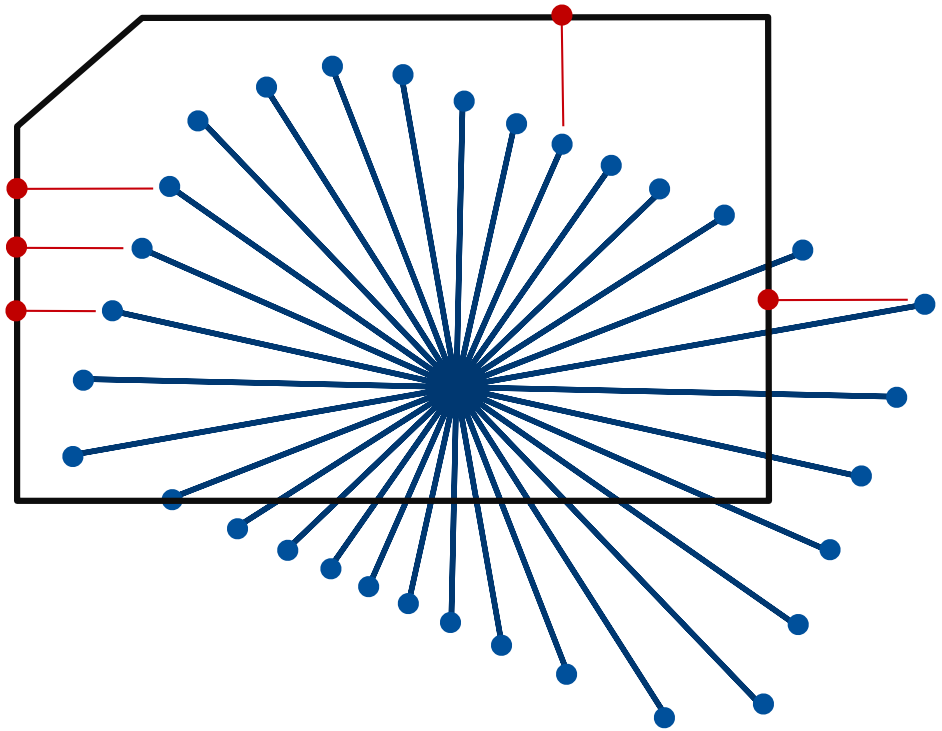
- May not work if the assumed (approximate) position/ orientation is very different from the real values

Fine localization



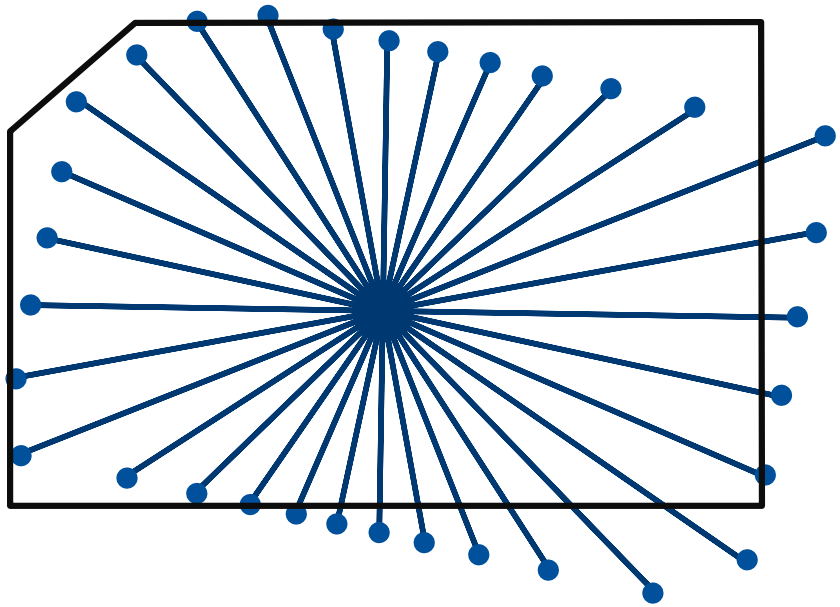
Fine localization

- Point correspondences (ICP): e.g. uses the closest point on a segment of the floor plan



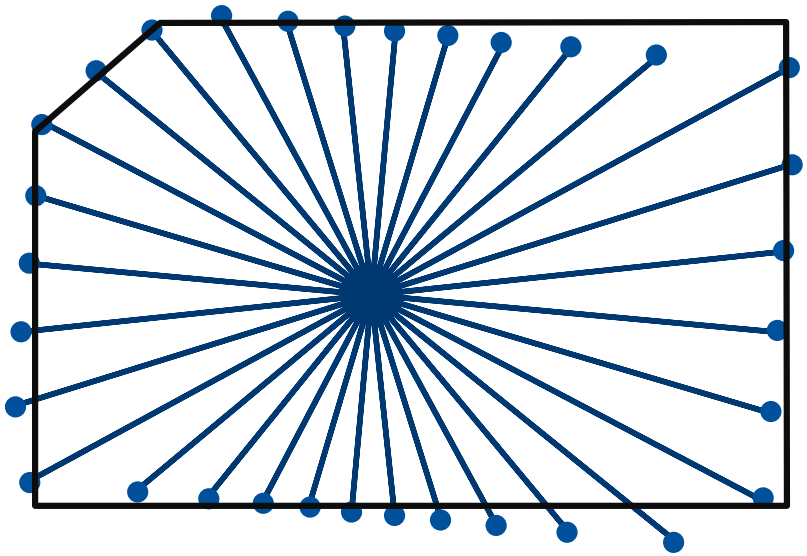
Fine localization

- Iteration…



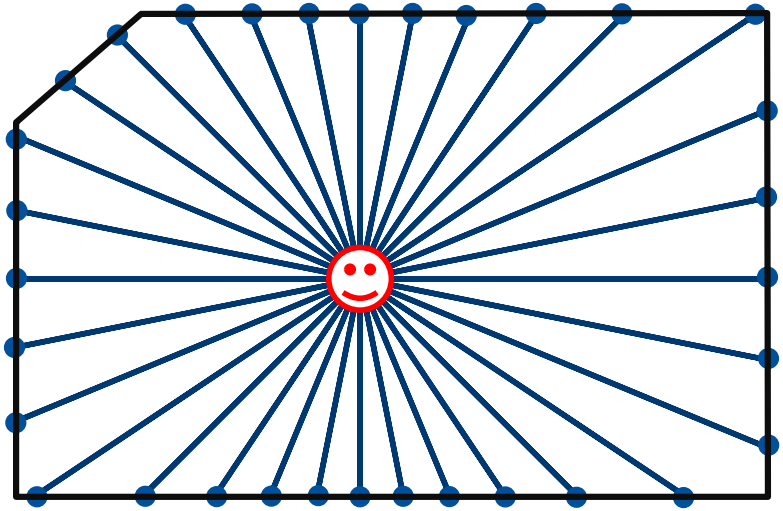
Fine localization

- Iteration…



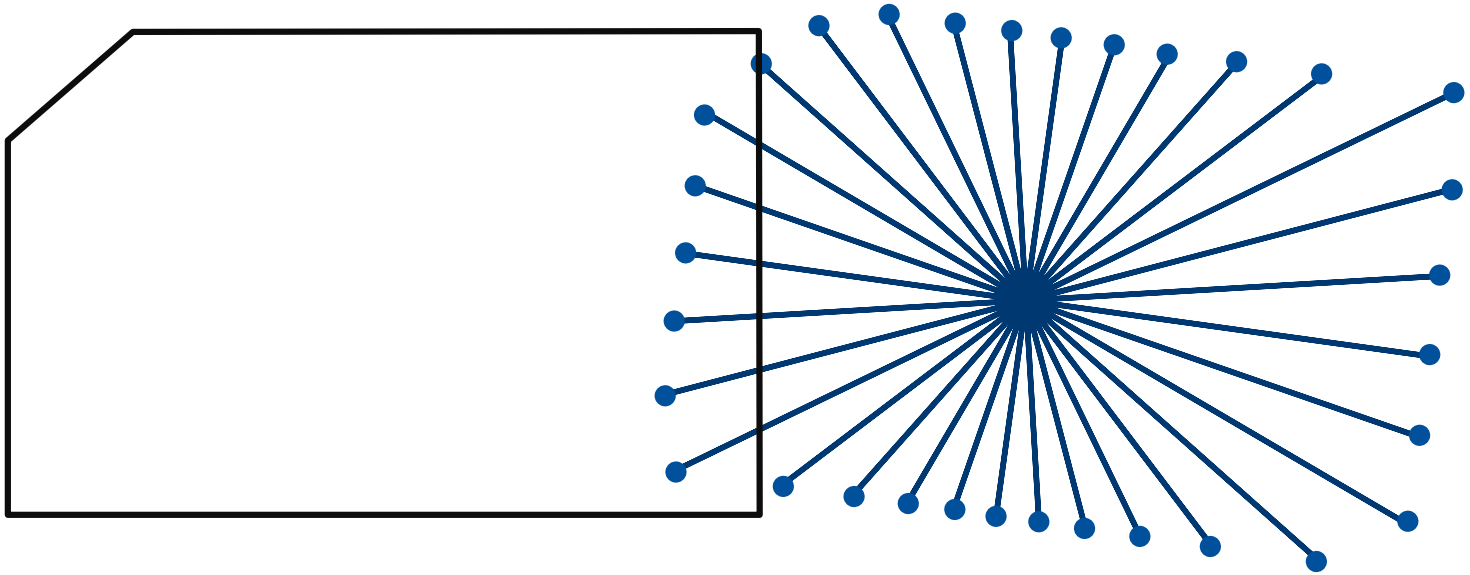
Fine localization

- Points aligned
- Also: found the position of the robot



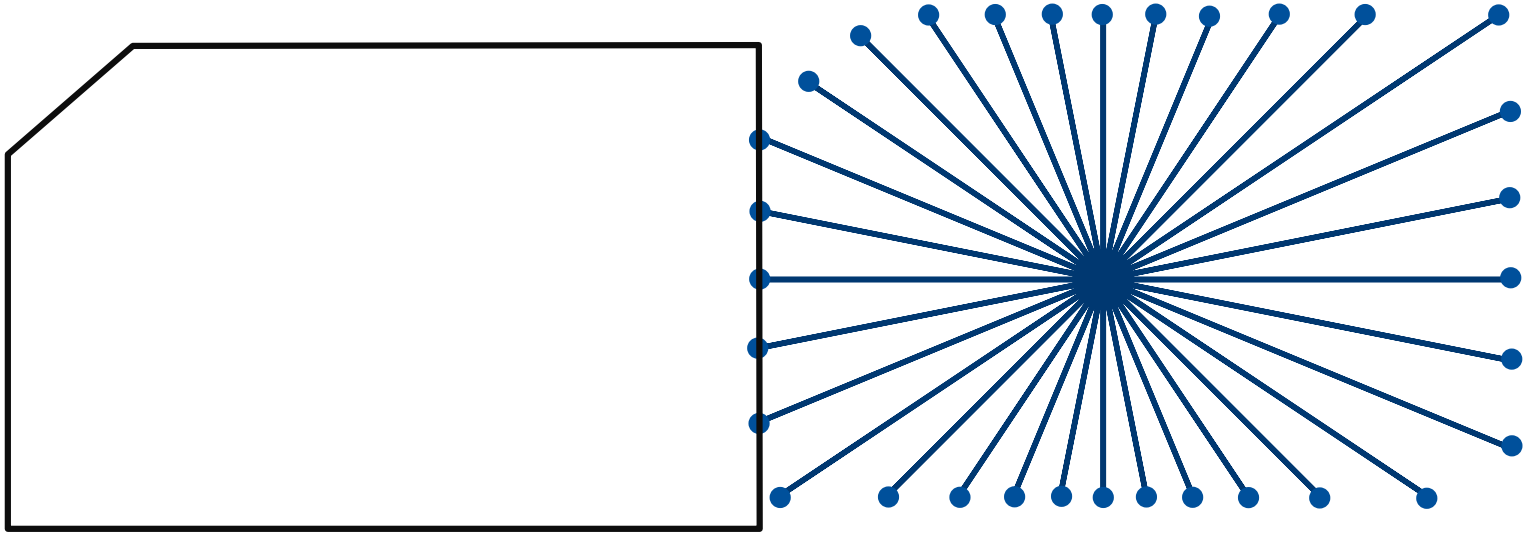
Coarse localization

- Fine registration / localization would not work in this case
  - There are no correspondences which would generate the correct solution



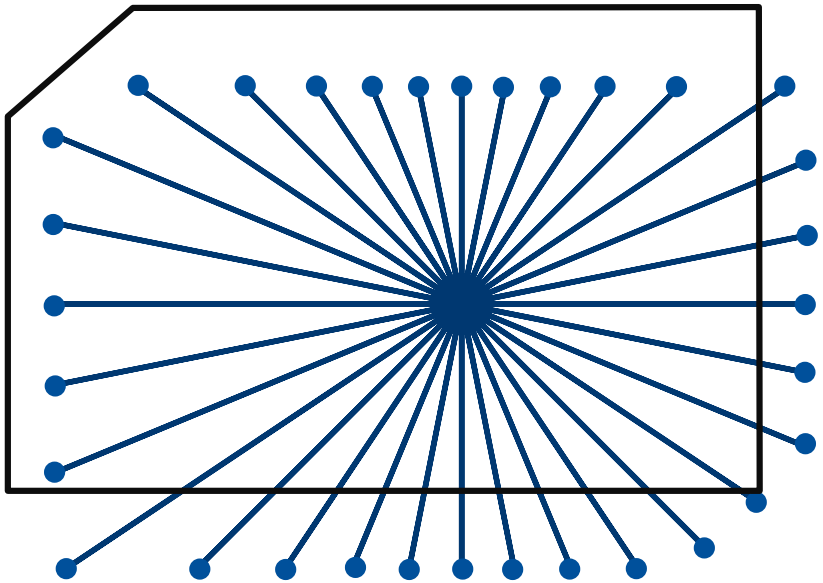
Coarse localization

- Fine registration / localization would not work in this case
  - There are no correspondences which would generate the correct solution



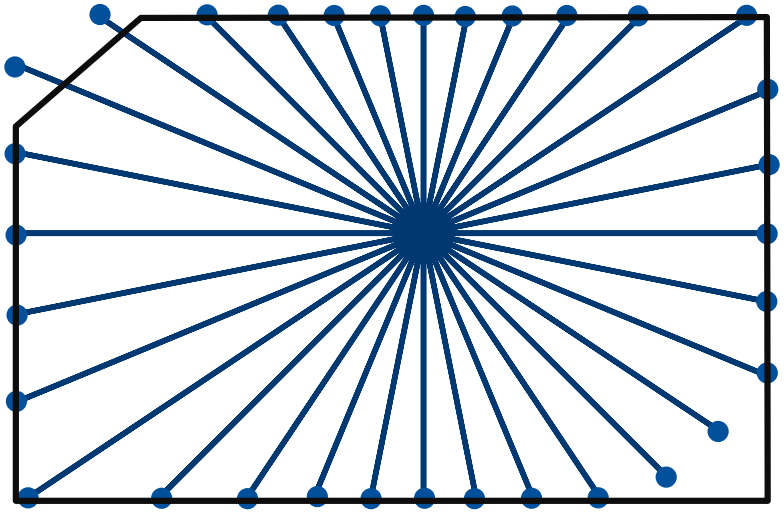
Coarse localization

- Fine registration / localization would not work in this case
  - Turned by 180° (typical problem of symmetries in the scene)



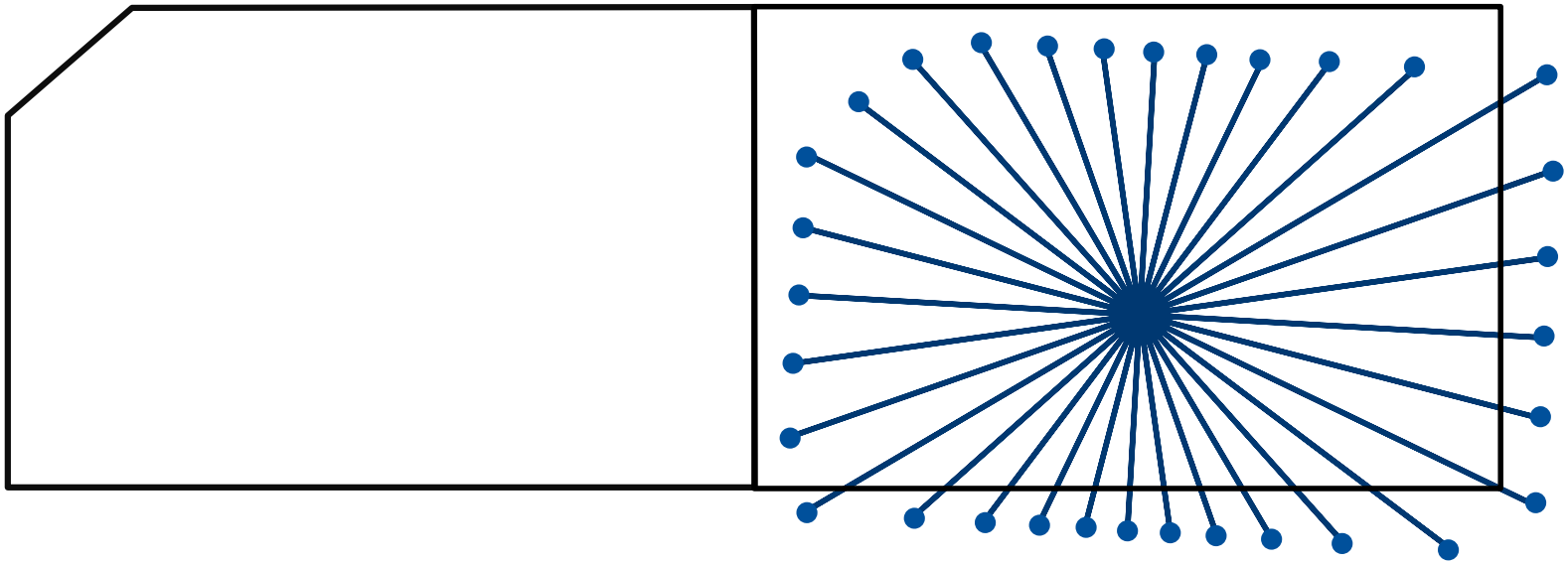
Coarse localization

- Fine registration / localization would not work in this case
  - Turned by 180° (typical problem of symmetries in the scene)



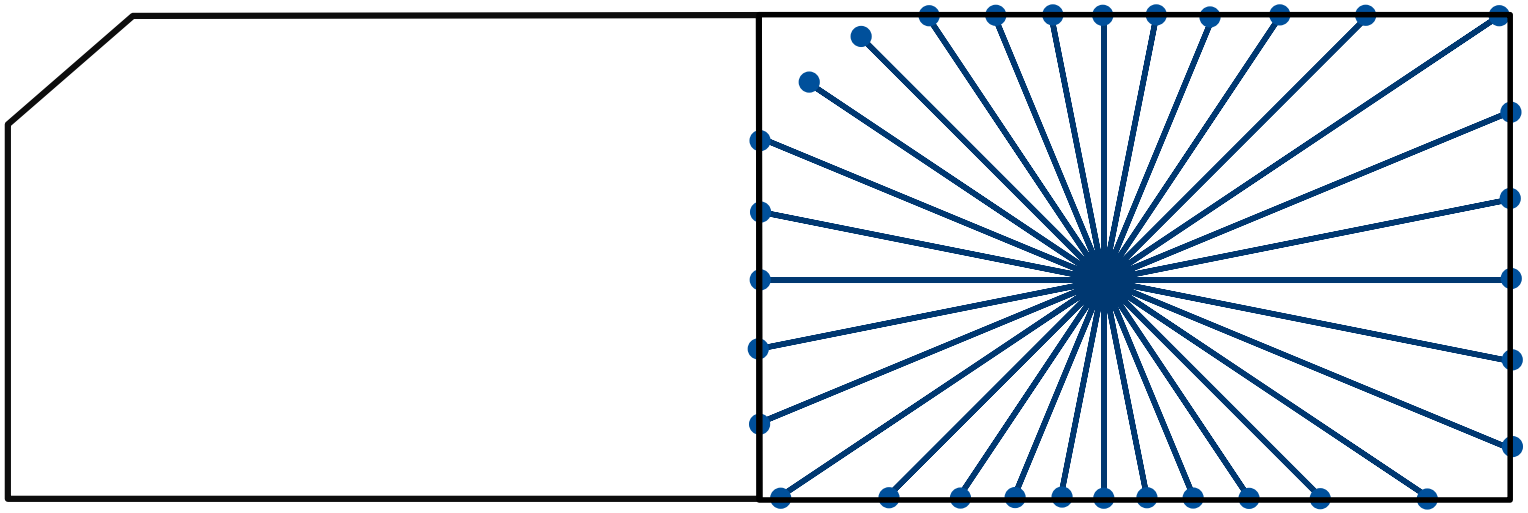
Coarse localization

- Fine registration / localization would not work in this case
  - Wrong global alternative



Coarse localization

- Fine registration / localization would not work in this case
  - Wrong global alternative



Coarse localization

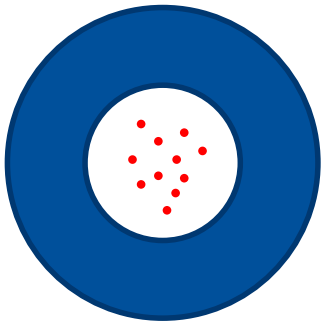
- Global (or at least “extended local”) search for a correct solution
- E.g. by assignment of features/ landmarks
- General approach – using the example of a given map

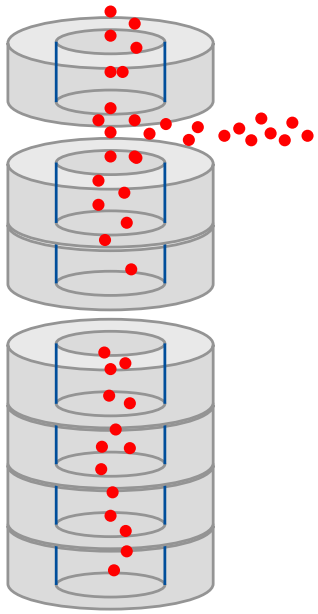
- Step 1: Offline
  - Extract certain features/ landmarks from the map
  - Put them into a database (georeferenced)

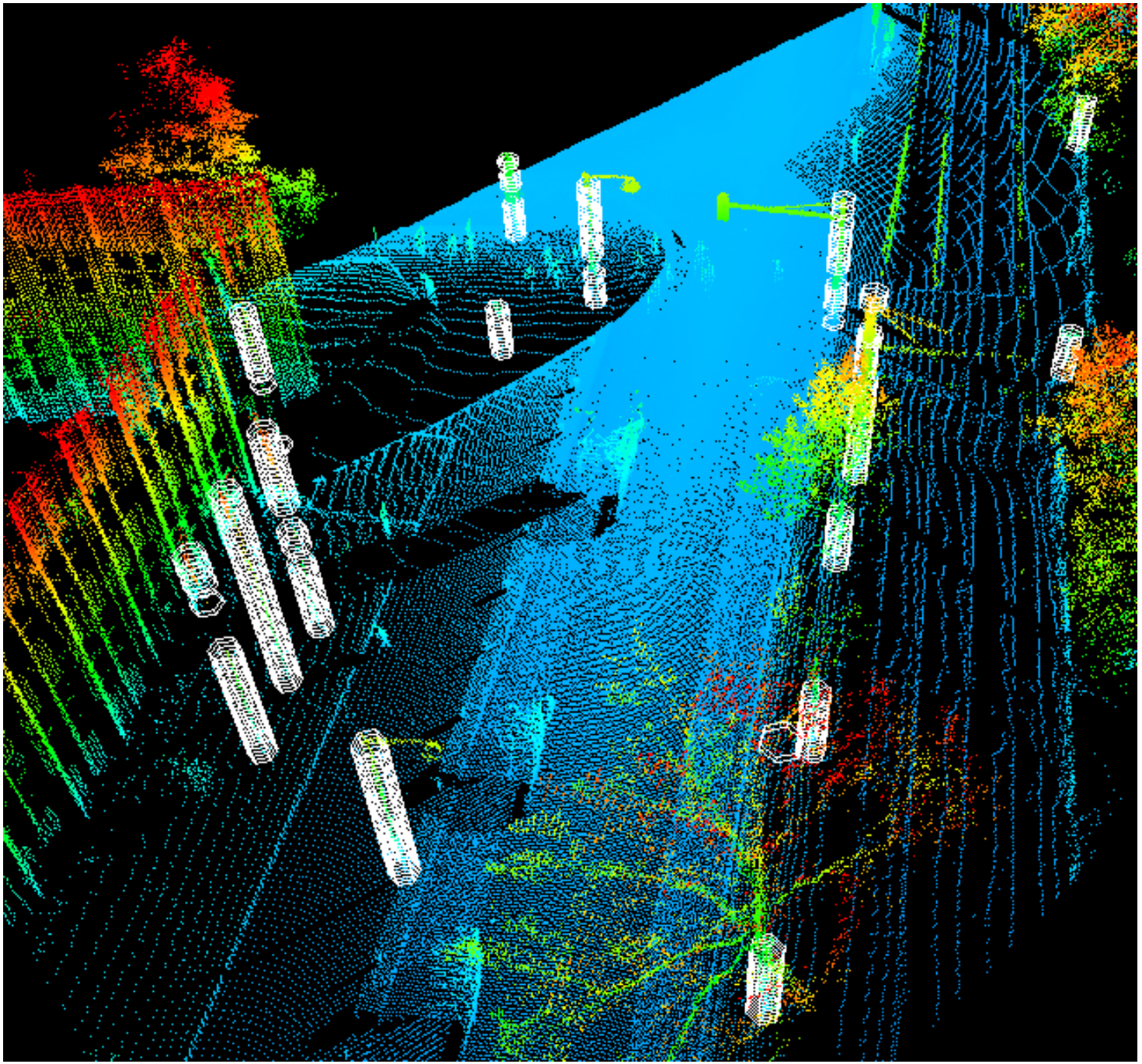
- Step 2: Online
  - From the measurement data of the robot/ vehicle, extract landmarks (normally, in real time!)
  - Assign those to the georeferenced landmarks in the map
  - This yields the current position

Example: assignment of points

Extraction: poles



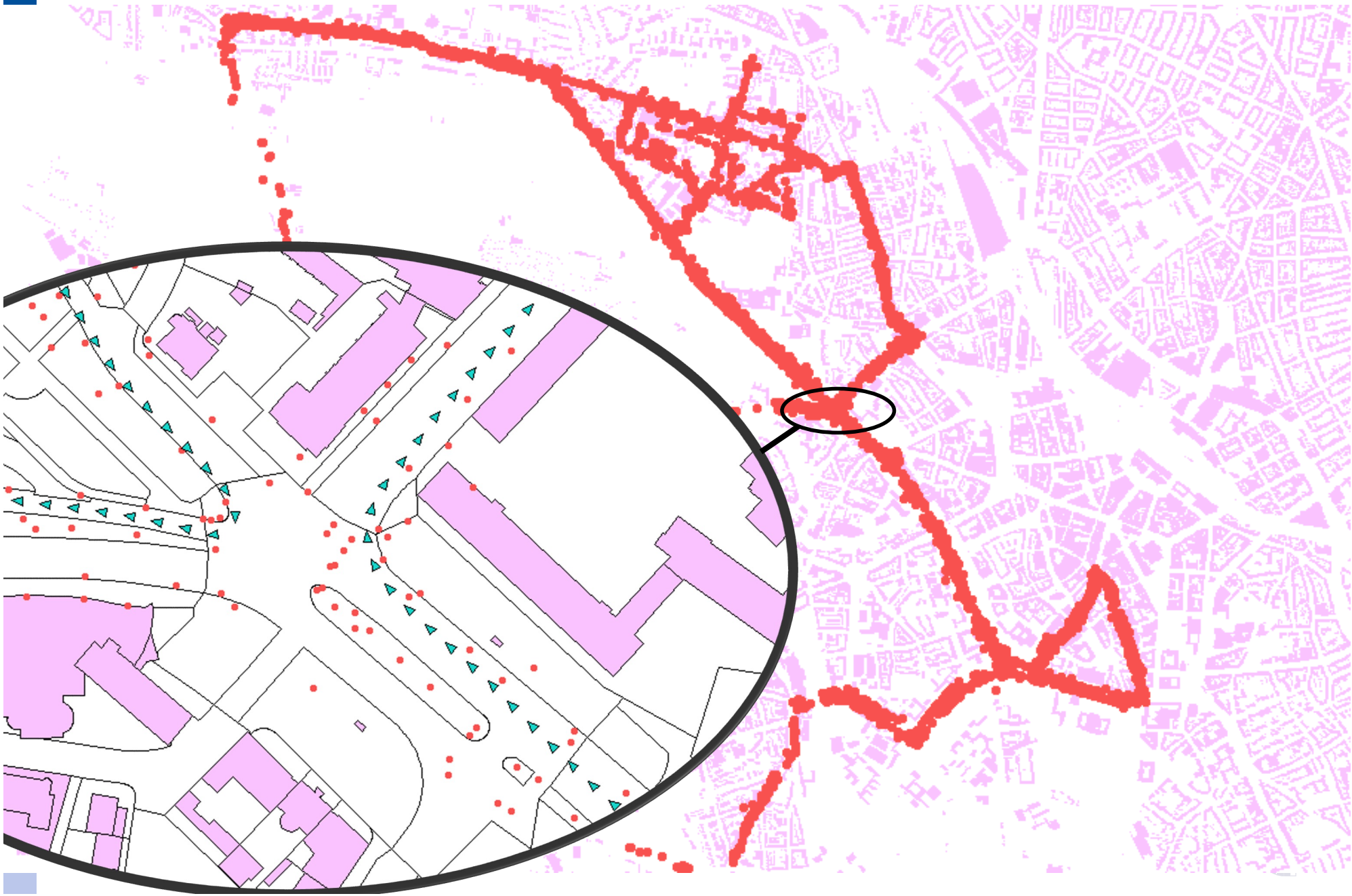




Assignment of points

- Offline:
  - Poles (street signs, traffic lights…), trees are extracted automatically from a LiDAR point cloud
  - From the point cloud of each object, a single point is obtained
    - E.g. center point, intersection of cylinder axis with ground
  - All those points are put into a database

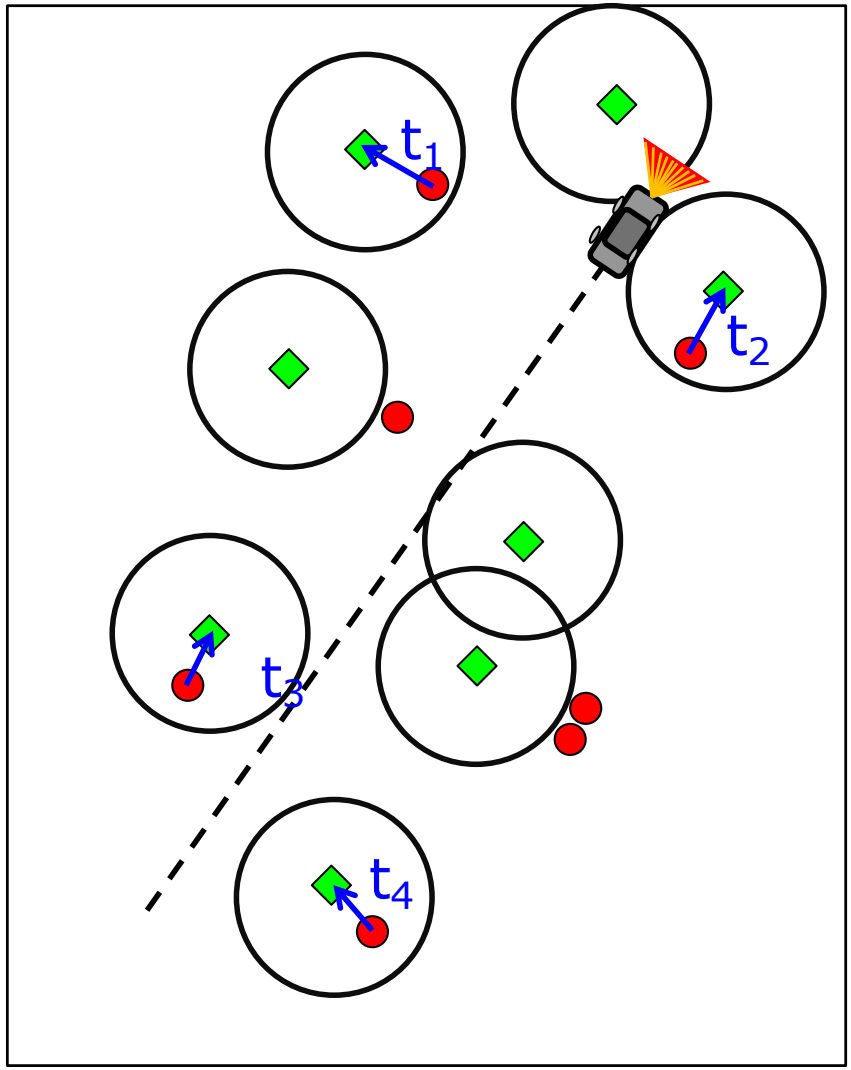
“Database of center points”

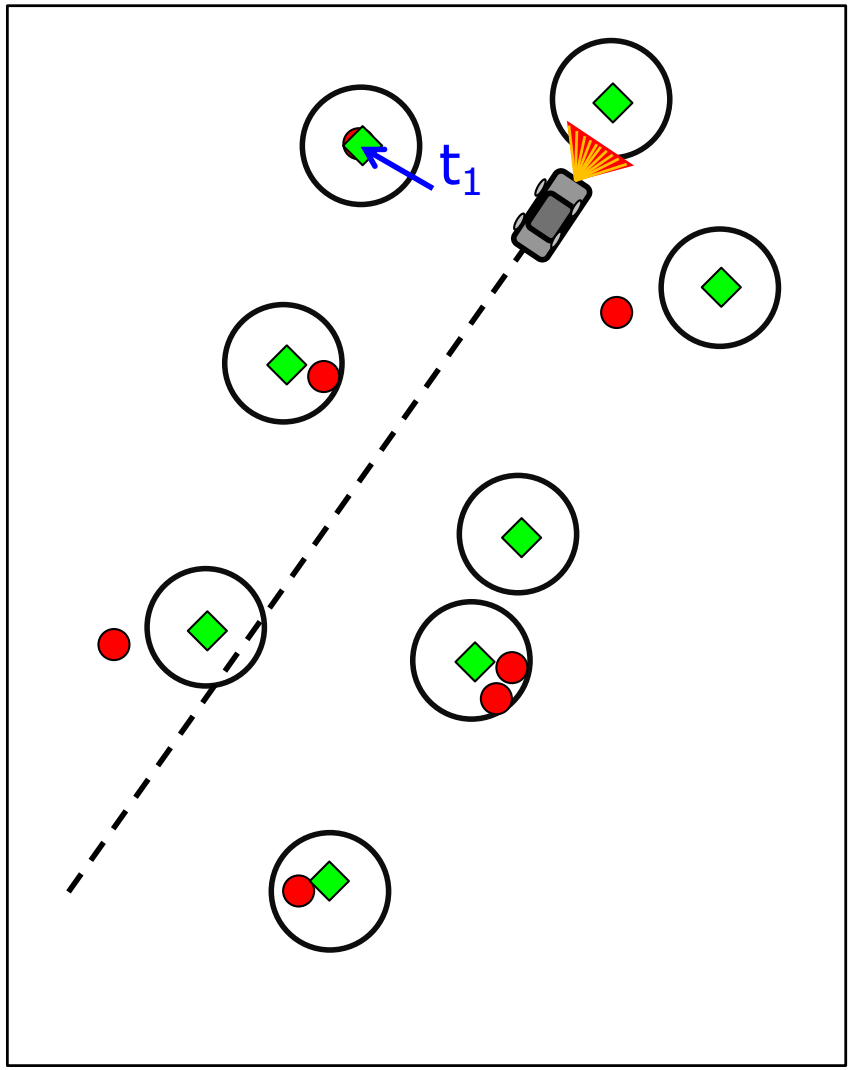


Assignment of points

- Online
  - Objects are extracted
  - From this, center points are computed as well
  - The local point pattern (around the vehicle) is matched to the database
  - Correspondences can be found e.g. using RANSAC

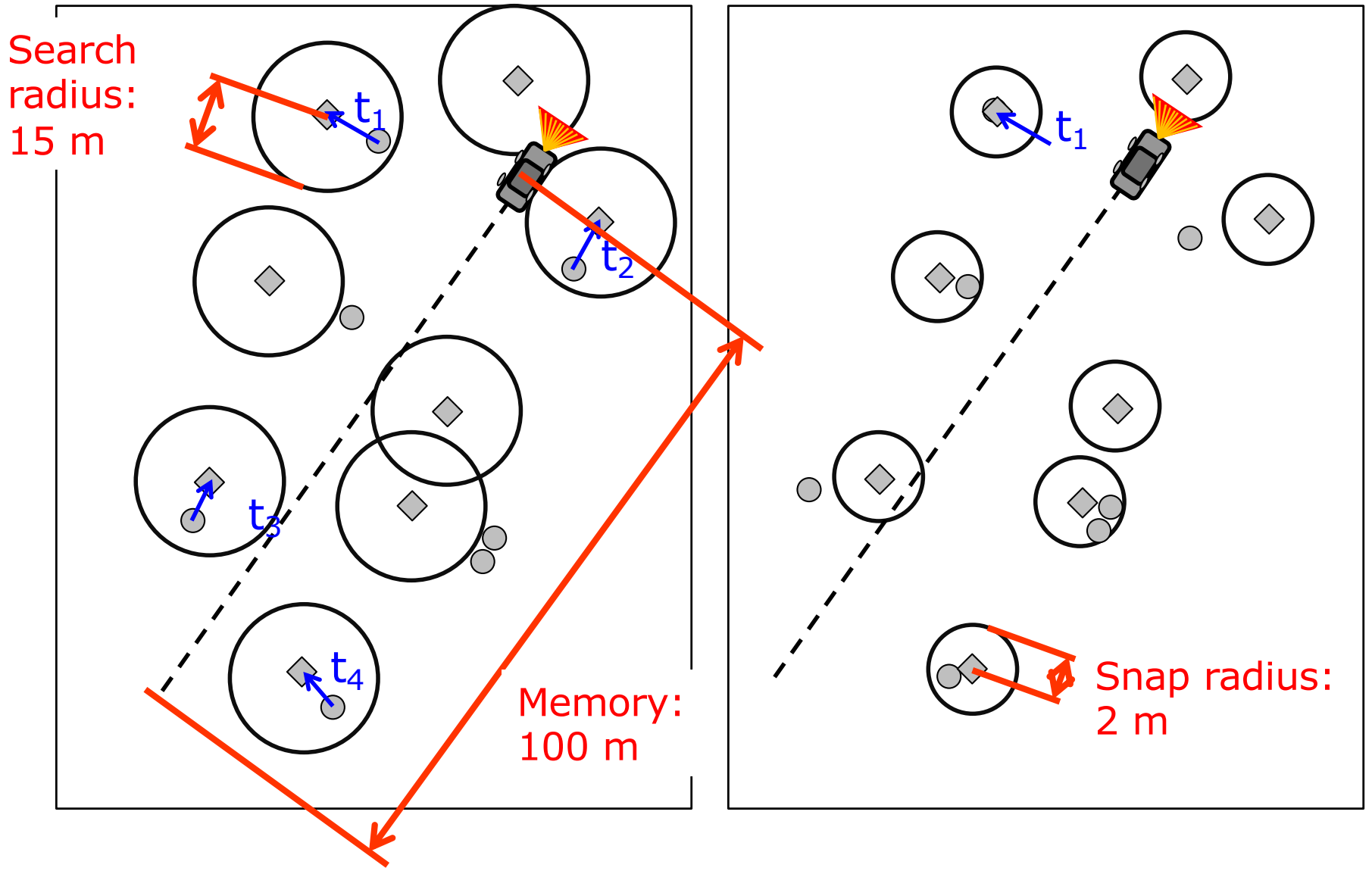
Position determination by assignment of landmarks (using RANSAC)





Poles detected by the vehicle
Reference poles from the map

Position determination by assignment of landmarks (using RANSAC)



poles_matching.mp4

Assignment of planes / straight line segments

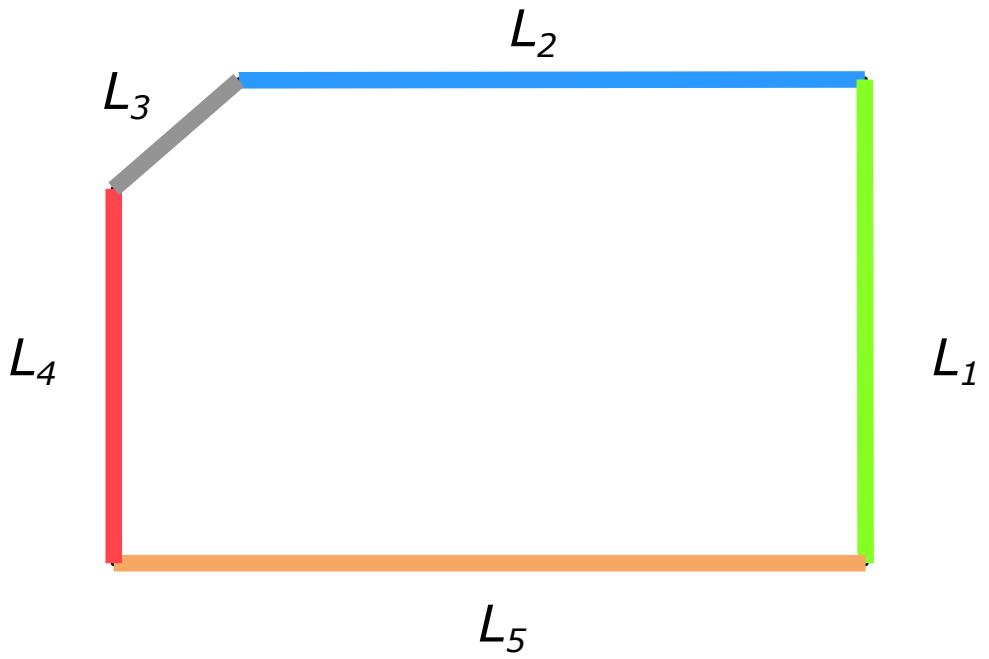
Step 1: Offline

- Extract features from the map

Map

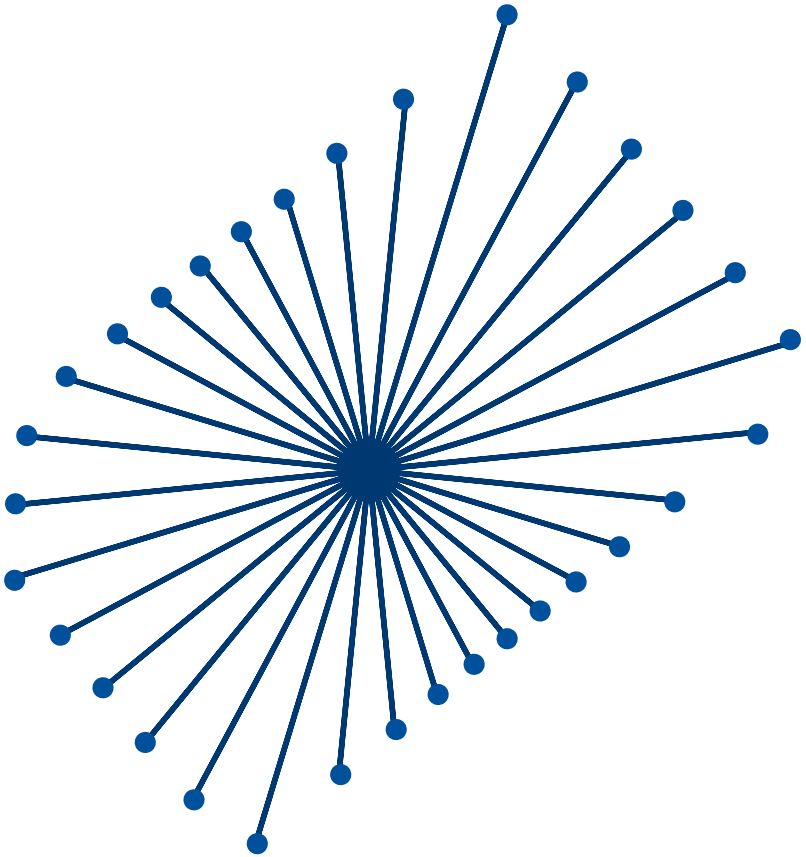
Step 1: Offline

- Extract features from the map
- In this case, very simple:
  - Model is given as a polygon
  - Features = Line segments of the polygon



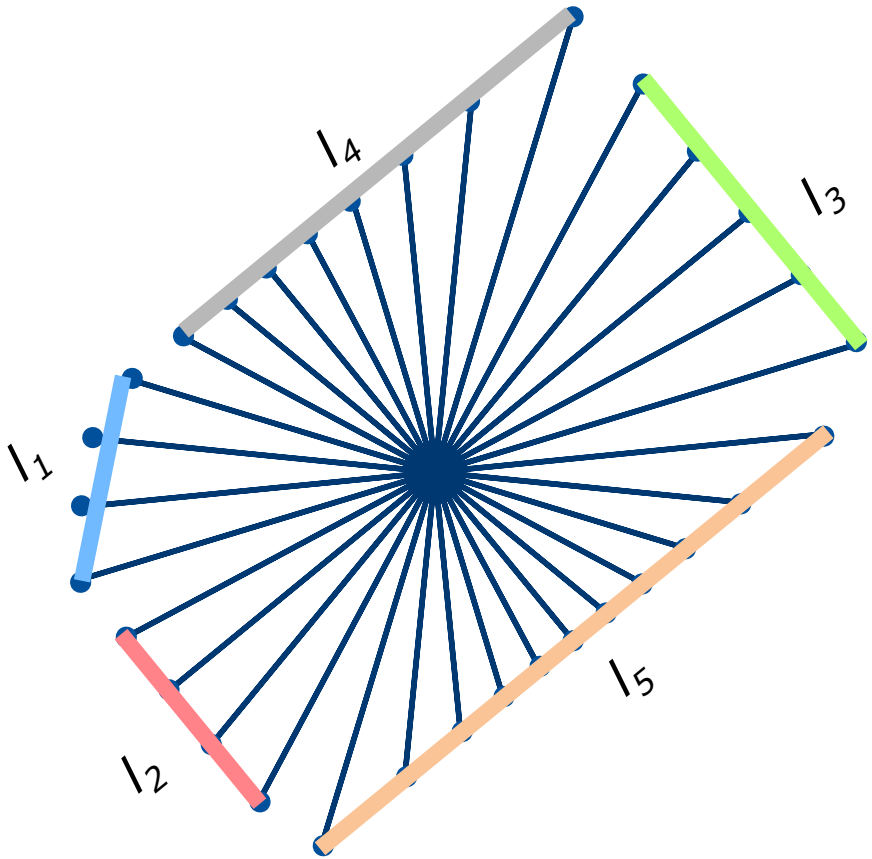
Step 2: Online

- Extract features from scan data
- Using point cloud segmentation



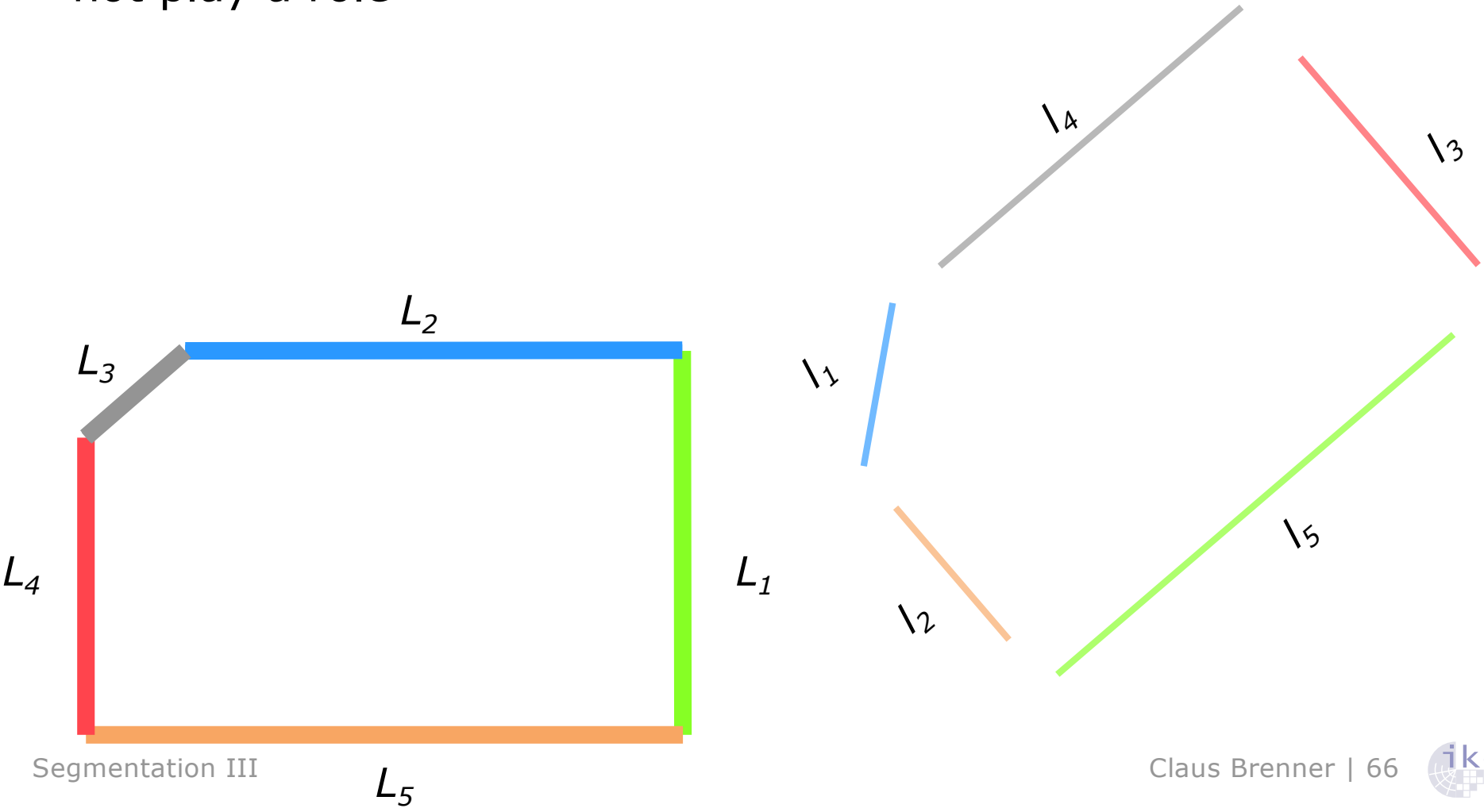
Step 2: Online

- Extract features from scan data
- Using point cloud segmentation



Assignment

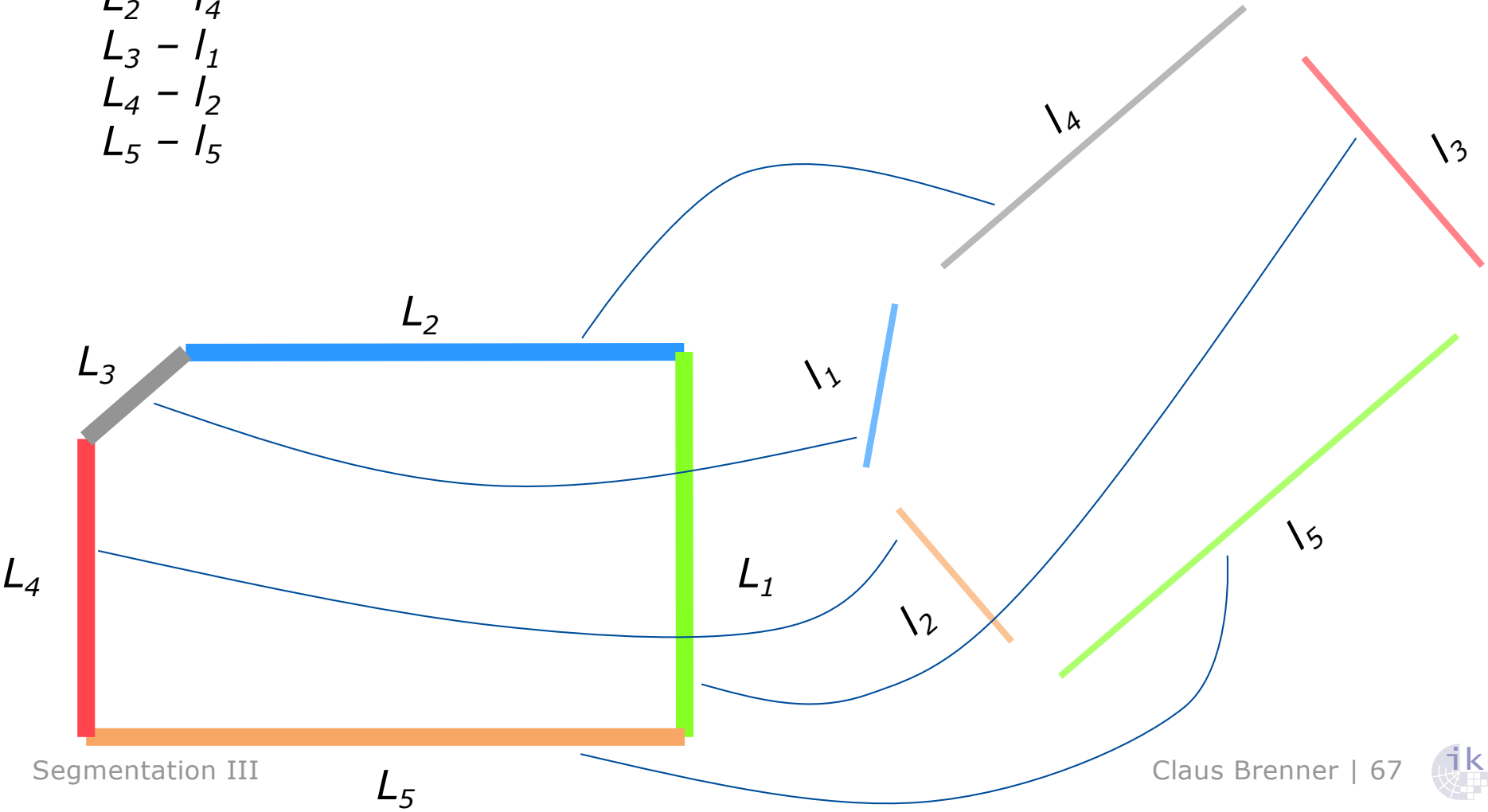
- Only features are assigned
- The original (scan) points do not play a role



Assignment

- „We“ know the correct assignment

$L_1 - l_3$
$L_2 - l_4$
$L_3 - l_1$
$L_4 - l_2$
$L_5 - l_5$

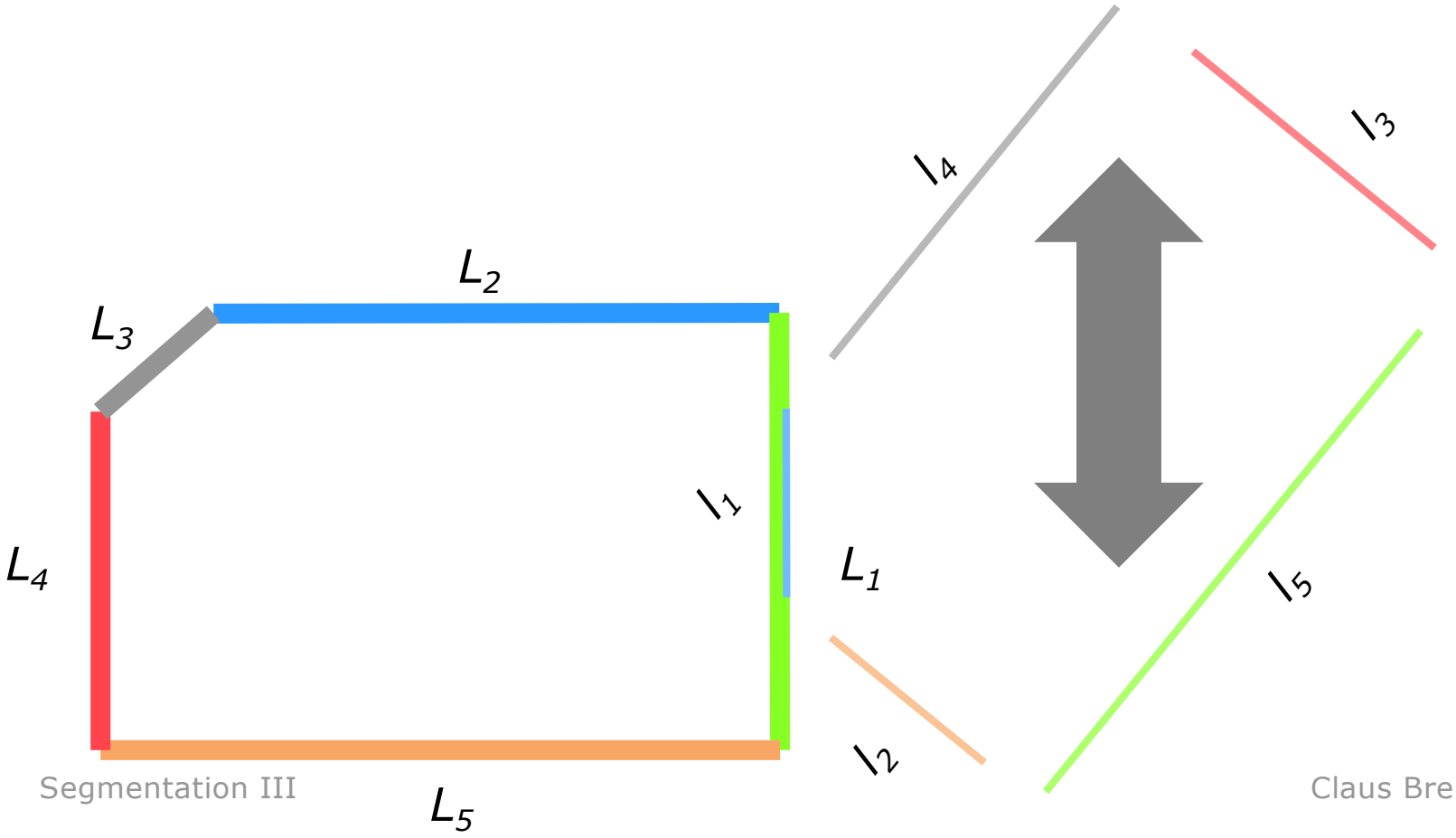


Assignment

- How can the computer find the correct assignment?
- Basic thoughts:
  - Transformation: rotation and translation. The scale is known.
    → 3 parameters
  - If **one** correspondence of segments is known, this will determine the rotation, but not the translation
  - If **two** correspondences of segments are known, rotation and translation are determined

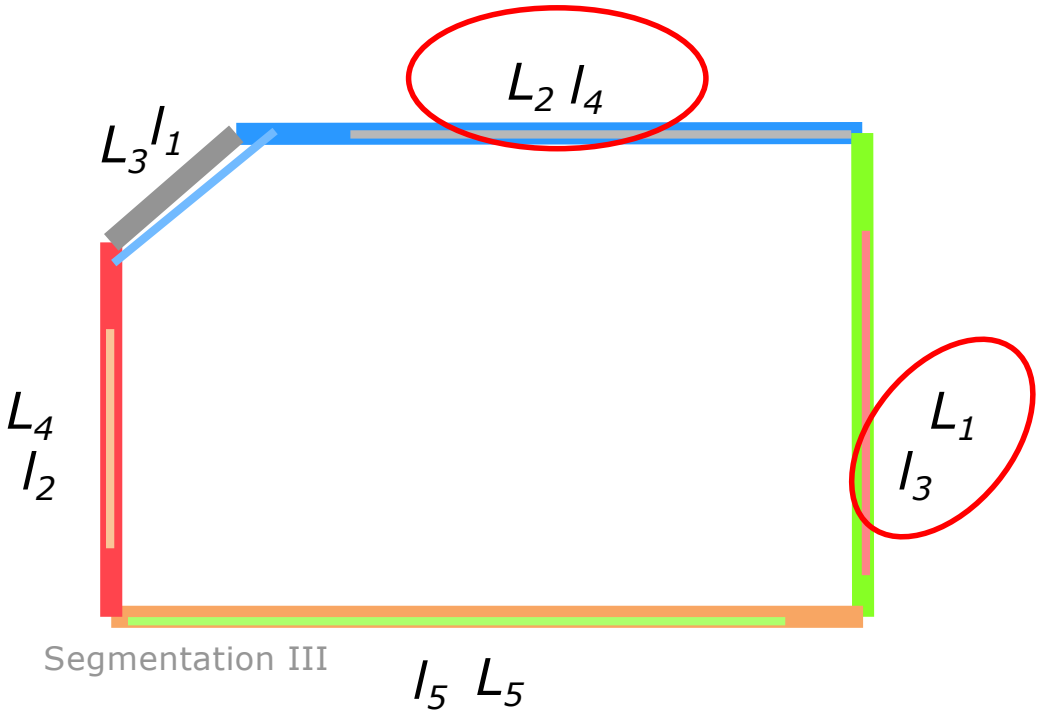
One line assignment

- E.g.: $L_1 - l_1$
- This determines rotation, but not translation
- A second line correspondence is needed to fix the translation



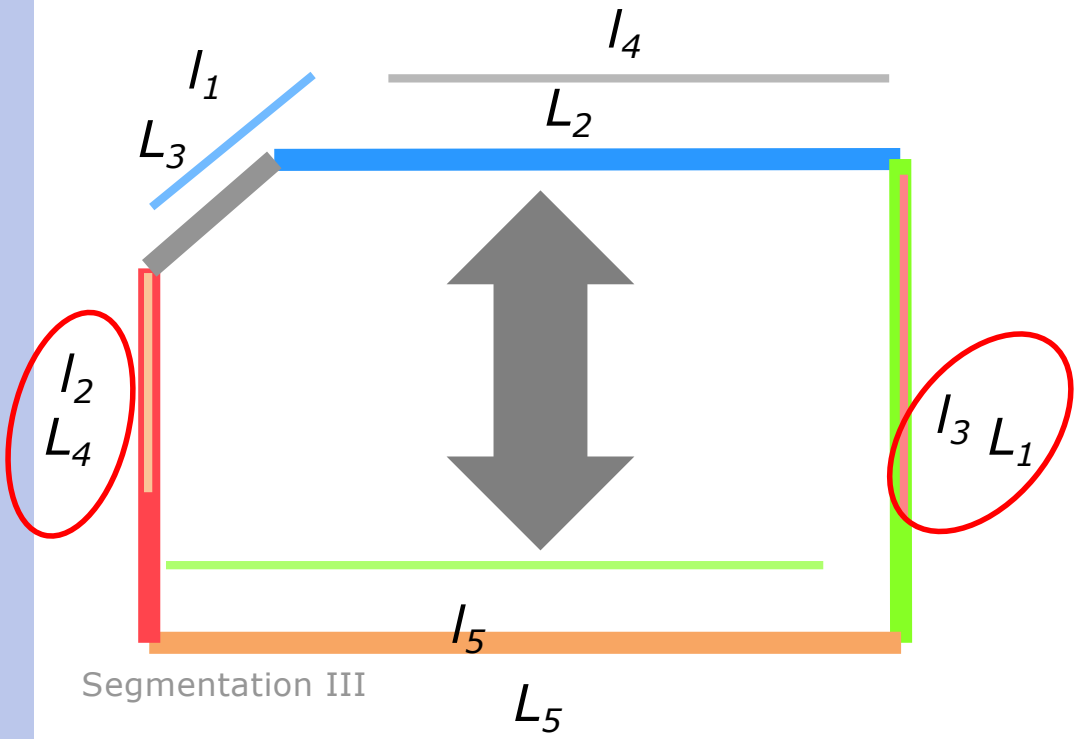
Two assigned line segments

- E.g.: L_2 – l_4 and L_1 – l_3
- This fixes rotation and translation



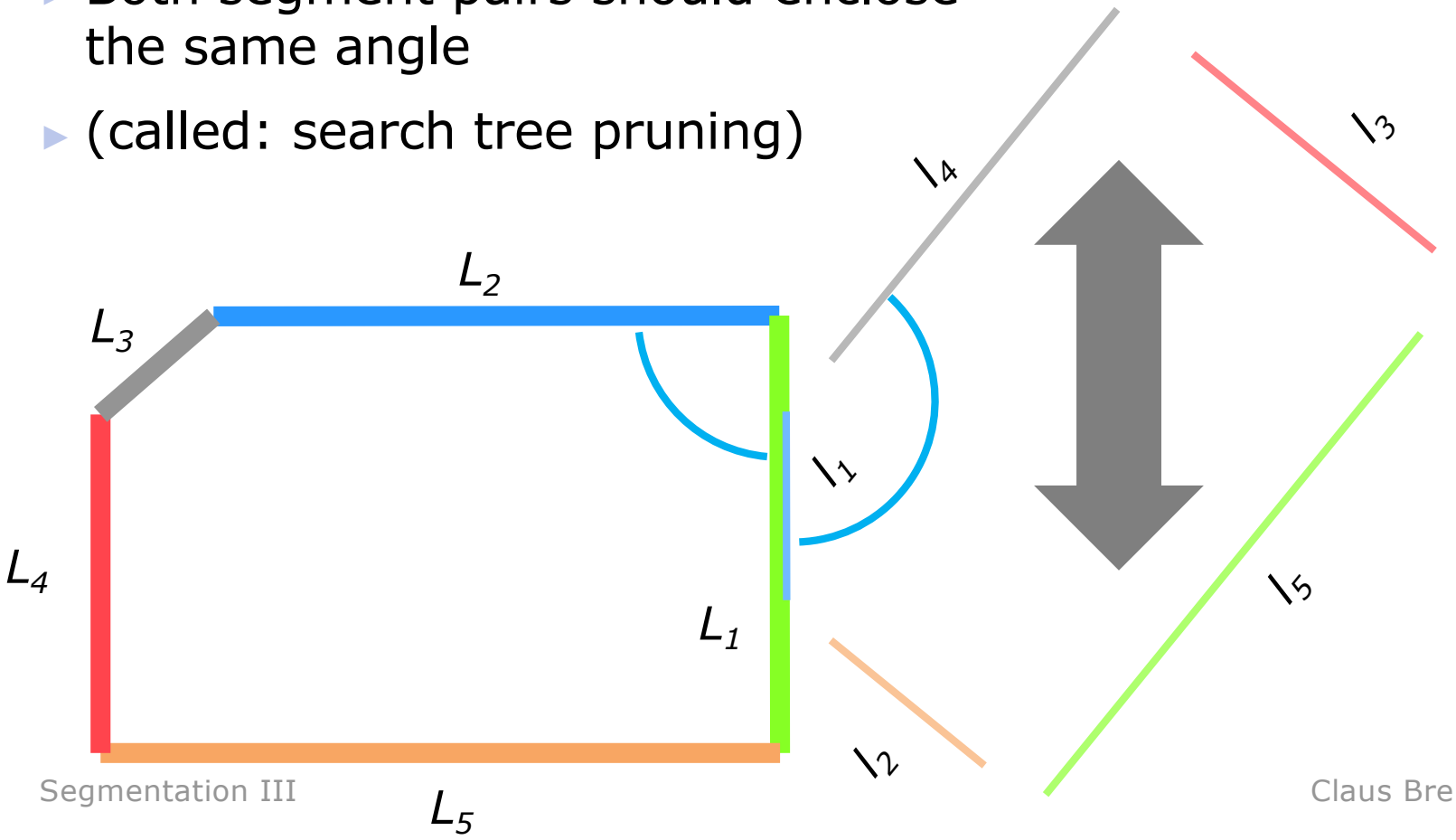
Two assigned line segments

- Observation 1: Correct, but unfavorable segment correspondences do not fix the translation
- E.g.: L_4 – l_2 and L_1 – l_3



Two assigned line segments

- Observation 2: Not all segment pairs make sense
- E.g. after picking L_1 – l_1 the combination L_2 – l_4 does not make sense
- Both segment pairs should enclose the same angle
- (called: search tree pruning)

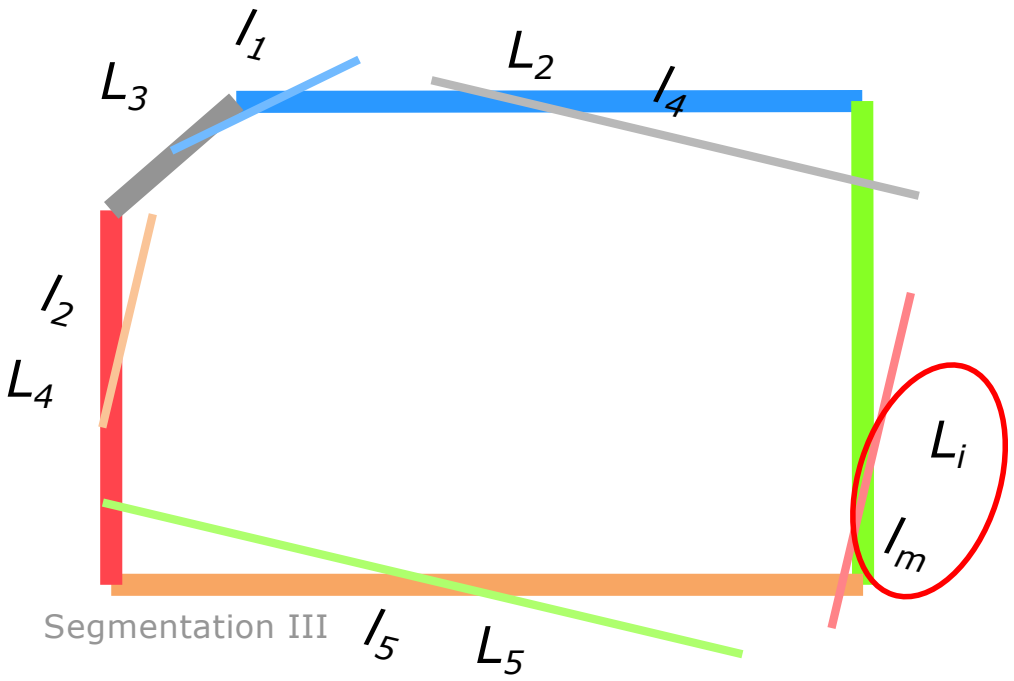


Example for an assignment algorithm

- Basic idea: RANSAC
- Repeat the following RANSAC experiment:
  - Draw two segments from the map: L_i, L_j
  - Draw two segments from the scene: l_m, l_n
  - Check the assignment $L_i - l_m$, $L_j - l_n$:
    - If *angle*(L_i, L_j) is not equal to *angle*(l_m, l_n) (using a certain tolerance), then discard this draw immediately
    - Otherwise: compute transformation and transform the scene accordingly
    - Get a score based on the overlapping lines
- In the end, choose the transformation with led to the highest score

Task 1: Compute transformation based on segment assignments

- E.g. the following algorithm can be used
- Part 1: Computing the rotation
  - Choose the first of the two line assignments: L_i – l_m
  - Compute the rotation matrix from those two segments (see the following slide)
- Part 2: Computing the translation



Part 1: computation of the rotation matrix from a segment pair

- Given: two segment equations, in HNF

$$L_i: \quad a_i x + b_i y + c_i = 0$$
$$l_m: \quad a_m x + b_m y + c_m = 0$$

- The normal vectors are (we assume unit length)

$$\mathbf{n}_i = \begin{bmatrix} a_i \\ b_i \end{bmatrix} \qquad \mathbf{n}_m = \begin{bmatrix} a_m \\ b_m \end{bmatrix}$$

- Then, the rotation matrix, which rotates $\mathbf{n}_i$ to $\mathbf{n}_m$ is given by:

$$\mathbf{R} = \begin{bmatrix} a_m & -b_m \\ b_m & a_m \end{bmatrix} \begin{bmatrix} a_i & b_i \\ -b_i & a_i \end{bmatrix} = \begin{bmatrix} a_m a_i + b_m b_i & a_m b_i - b_m a_i \\ b_m a_i - a_m b_i & b_m b_i + a_m a_i \end{bmatrix}$$

Part 2: computation of the translation from two segment pairs

- Segment equations in HNF:

$$L_i: \quad a_i x + b_i y + c_i = \langle \mathbf{R}\mathbf{n}_i, \mathbf{x} \rangle + c_i = 0 \qquad L_j: \quad \langle \mathbf{R}\mathbf{n}_j, \mathbf{x} \rangle + c_j = 0$$
$$l_m: \quad a_m x + b_m y + c_m = \langle \mathbf{n}_m, \mathbf{x} \rangle + c_m = 0 \qquad l_n: \quad \langle \mathbf{n}_n, \mathbf{x} \rangle + c_n = 0$$

- The rotation was determined such that:

$$\mathbf{R}\mathbf{n}_i = \mathbf{n}_m \quad \text{and} \quad \mathbf{R}\mathbf{n}_j = \mathbf{n}_n$$

- Now, we are looking for the translation **t**, which fulfills:

$$\left.\begin{aligned} \langle \mathbf{n}_m, \mathbf{x} + \mathbf{t} \rangle + c_i &= 0 \\ \langle \mathbf{n}_m, \mathbf{x} \rangle + c_m &= 0 \end{aligned}\right\} \qquad \left.\begin{aligned} \langle \mathbf{n}_n, \mathbf{x} + \mathbf{t} \rangle + c_j &= 0 \\ \langle \mathbf{n}_n, \mathbf{x} \rangle + c_n &= 0 \end{aligned}\right\}$$

Part 2: computation of the translation from two segment pairs

- By subtracting the two equation systems, one obtains

$$\langle \mathbf{n}_m, \mathbf{t} \rangle = c_m - c_i \qquad \langle \mathbf{n}_n, \mathbf{t} \rangle = c_n - c_j$$

- Or (using the definition of the scalar product)

$$\begin{aligned} \mathbf{n}_m^{\mathrm{T}} \mathbf{t} &= c_m - c_i \\ \mathbf{n}_n^{\mathrm{T}} \mathbf{t} &= c_n - c_j \end{aligned}$$

- Or, in matrix notation

$$\begin{bmatrix} \mathbf{n}_m^{\mathrm{T}} \\ \mathbf{n}_n^{\mathrm{T}} \end{bmatrix} \mathbf{t} = \begin{bmatrix} c_m - c_i \\ c_n - c_j \end{bmatrix}$$

Part 2: computation of the translation from two segment pairs

- From this we obtain the following solution

$$\mathbf{t} = \begin{bmatrix} \mathbf{n}_m^{\mathbf{T}} \\ \mathbf{n}_n^{\mathbf{T}} \end{bmatrix}^{-1} \begin{bmatrix} c_m - c_i \\ c_n - c_j \end{bmatrix}$$

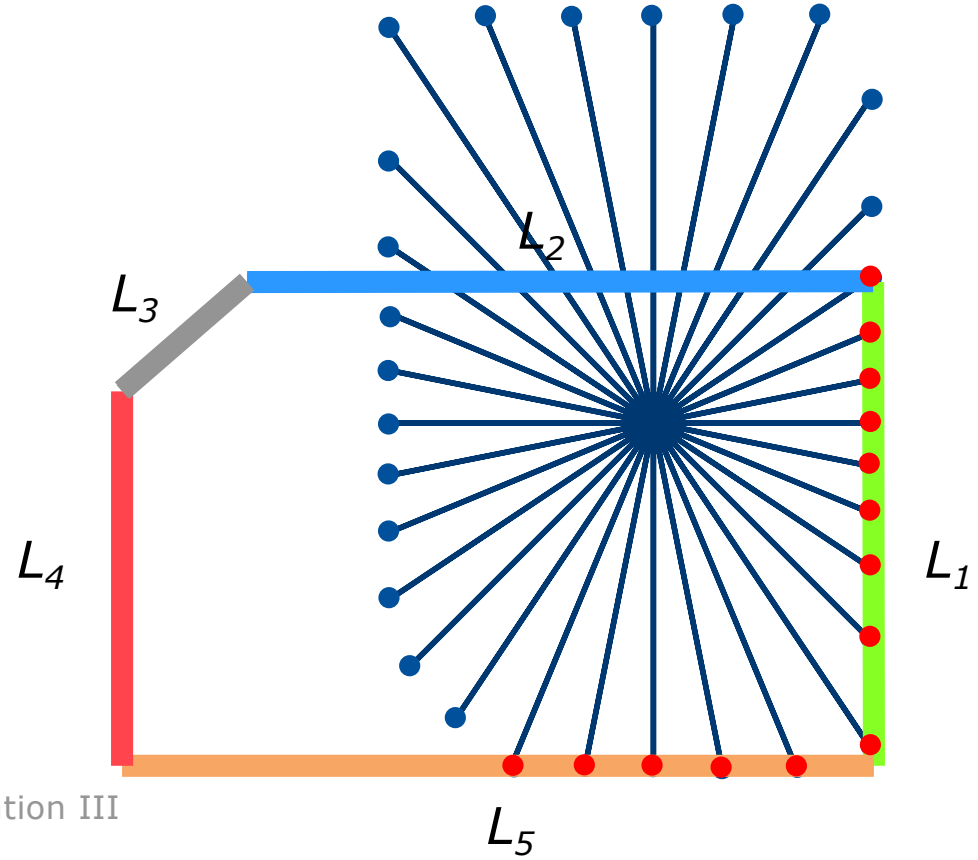
- Now, we have determined both, the rotation and the translation
- From this solution, we immediately see that if the selected line pairs are parallel, then

$$\begin{bmatrix} \mathbf{n}_m^{\mathbf{T}} \\ \mathbf{n}_n^{\mathbf{T}} \end{bmatrix}$$

is singular, i.e. **t** cannot be determined in this case.

Task 2: Compute the score

- The score can be computed by
  - using the computed transformation to transform all original scan points and then
  - counting all points which are close to a line in the map



(Alternatively, the overlap can be determined from the overlap of the segments)

Final remark: transformation of lines

- Assume the transformation of points is given by (e.g. affine):

$$\mathbf{x}' = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \mathbf{x} + \begin{bmatrix} e \\ f \end{bmatrix}$$

- Or, in homogeneous notation:

$$\mathbf{x}'_h = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{H}\mathbf{x}_h$$

- Then, the transformation of a line $\mathbf{l}$: $ax + by + c = 0$ is given by

$$\mathbf{l}' = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = (\mathbf{H}^{-1})^{\mathrm{T}} \mathbf{l} = \mathbf{H}^{-\mathrm{T}} \mathbf{l}$$

(and vice versa)

Literature

- X. Jiang and H. Bunke. Fast segmentation of range images into planar regions by scan line grouping. Machine Vision and Applications, 7(2): 115-122, 1994.
- C. Brenner, C. Dold, N. Ripperda. Coarse orientation of terrestrial laser scans in urban environments. ISPRS Journal of Photogrammetry & Remote Sensing 63 (2008), 4–18.
- Brenner, C., 2010. Vehicle localization using landmarks obtained by a LiDAR mobile mapping system. In: Paparoditis N., Pierrot-Deseilligny M., Mallet C., Tournaire O. (Eds), IAPRS, Vol. XXXVIII, Part 3A – Saint-Mandé, France, September 1-3, 2010, 139–144.