Institut für Kartographie und Geoinformatik | Leibniz Universität Hannover

Segmentation II

Claus.Brenner@ikg.uni-hannover.de

Leibniz Universität Hannover

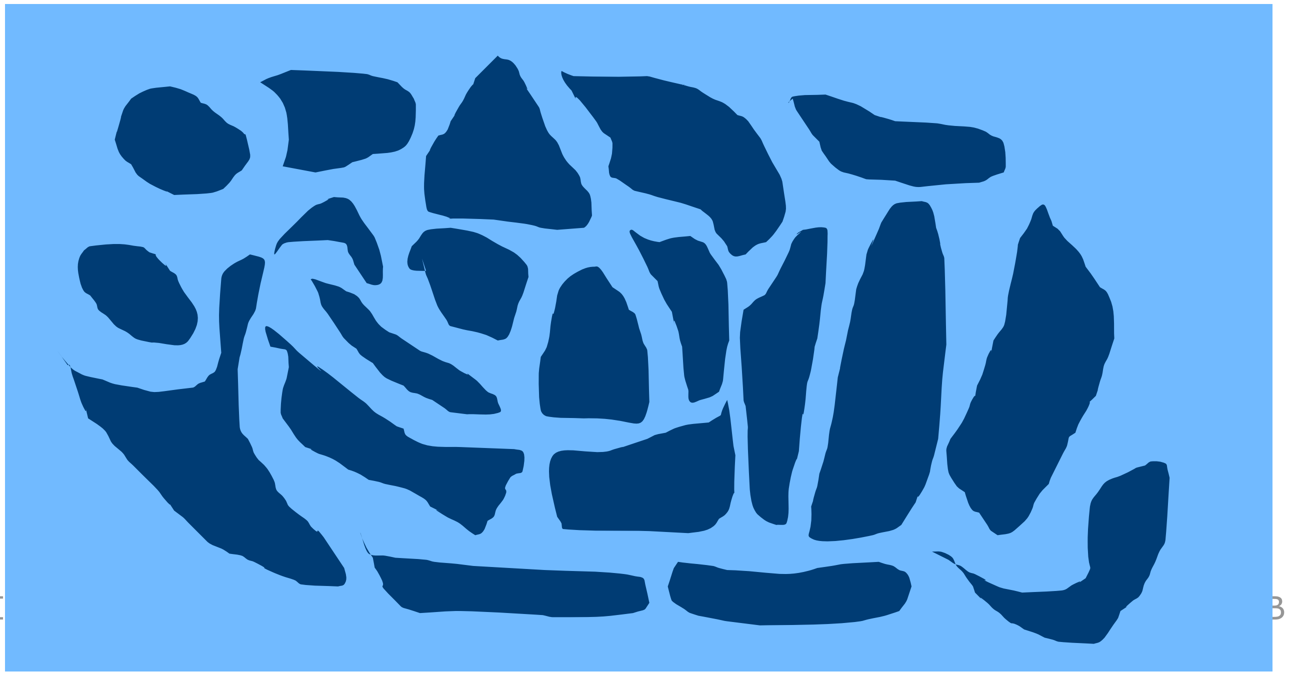
RANSAC: Optimization

Computational complexity of RANSAC

- Complexity of RANSAC depends on:
  1. How many draws (*random samples*) are necessary
  2. How costly it is to evaluate each draw (determination of the *consensus set*)

Example: number of draws

- Especially if RANSAC is used for segmentation, *w* can be very small
- E.g. we want to find all planar segments in a scene
- Assume:
  - 50% of the data cannot be assigned to a segment
  - The remaining data in the scene consists of 20 different planar segments



Example: number of draws

- With prob. 1/2 a point is inside an (arbitrary) segment
- With prob. $(1/40)^2$, two more random draws lead to points inside the same segment
- Then, $b = 1/2 * (1/40)^2 = 0.0003125$. For $z=0.99$ we get:

$$\boldsymbol{k = 14735}$$

- In general, we have to optimize how we draw the points, so that the probability to get a good draw is high.

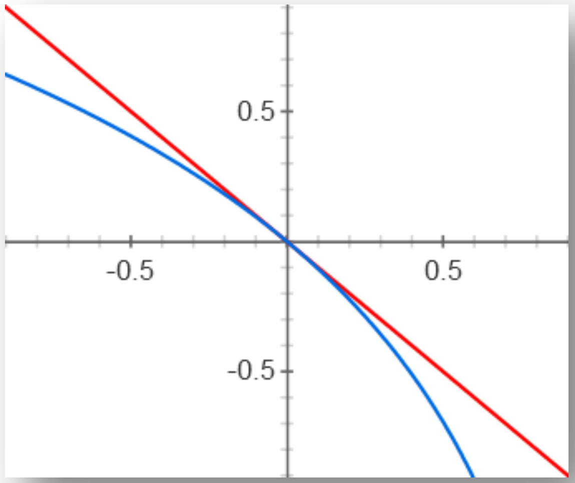
Number of draws

- Revisit the calculation of the number of draws:

$$k > \frac{\ln(1-z)}{\ln(1-b)}$$

- Where *b* is the probability that a draw (which consists of several draws of observations) belongs to the model
- For small *b* the following holds:

$$\ln(1-b) \approx -b$$



- Thus:

$$k_{\min} \approx \frac{-\ln(1-z)}{b}$$

- I.e., for small *b* (since *z* is fixed)

$$k_{\min} \propto 1/b$$

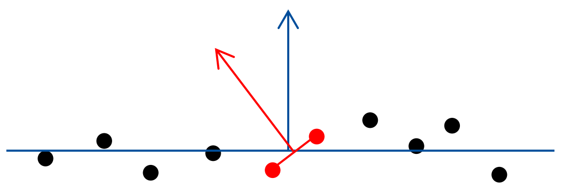
Number of draws

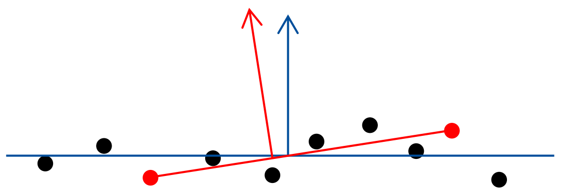
- The cost of RANSAC is (approximately) inversely proportional to the probability of a good draw (i.e.: if the probability of a good draw halves, the number of necessary draws doubles)
- If the determination of the *consensus set* has cost *C*, then it follows:
  - RANSAC has cost: $O(C/b)$
- In the previous example:
  - Assume that the point cloud to be segmented has 1 M points (not very many by today's standards)
  - 14735 times, a plane has to be determined from 3 points
  - For **each** of these planes, 1 M point-to-plane distances have to be computed
    $\rightarrow$ ~ 15 billion distance computations.

RANSAC: Optimization 1: Required number of draws

Optimization of the number of draws if object properties / data are known

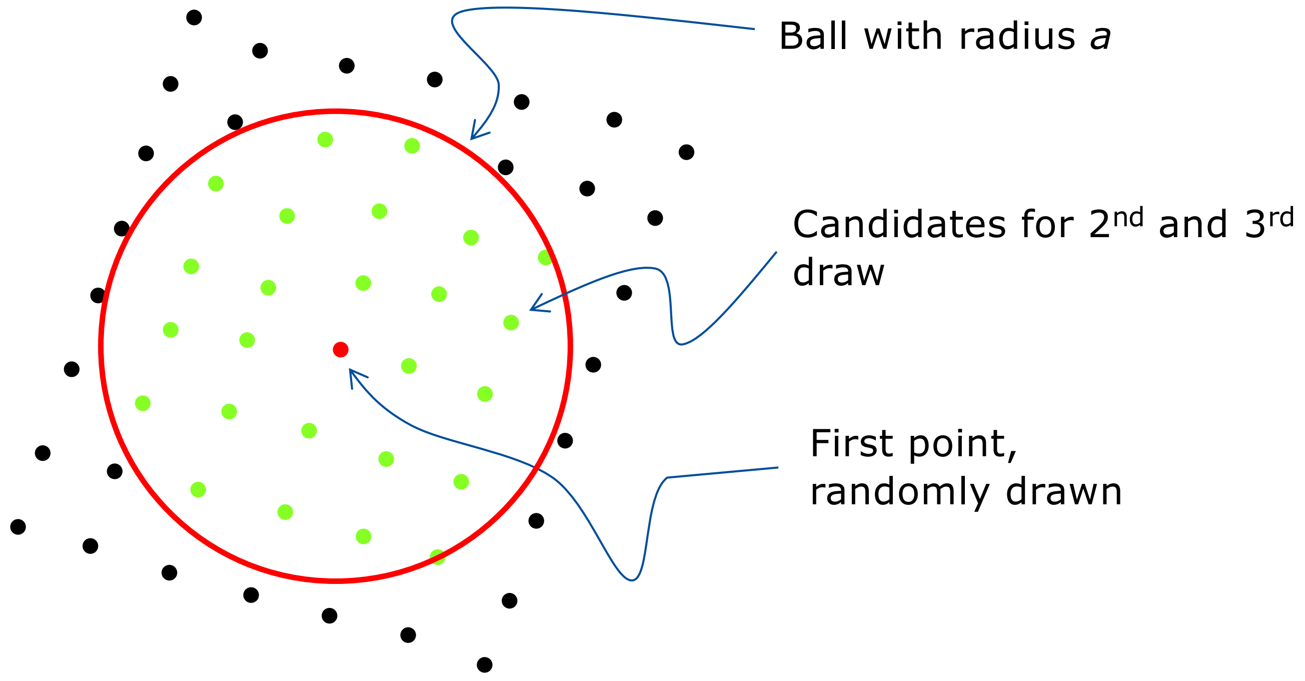
- Sometimes, known properties about the object space or the characteristics of the data can be used for optimization
- Example: Scan of a room, segmentation into planar patches (walls, floor, ceiling)
  - Assumption 1: The patches are expected to be at most a few meters in size (e.g. room height). = *Knowledge about object space.*
  - Assumption 2: Scanner noise is limited. If three points of a planar segment are selected which are (e.g.) at least ~10 centimeters apart, the plane equation can be determined = *Knowledge about the characteristics of the data.*





Optimization of the number of draws if object properties / data are known

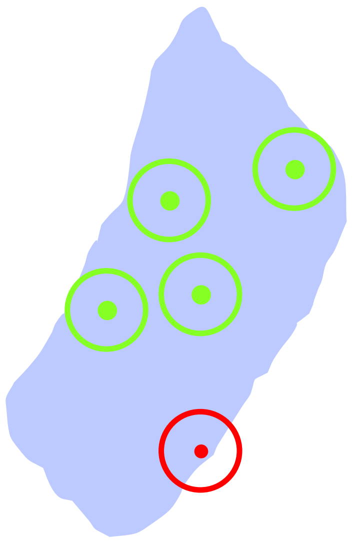
- Then: Instead of 3 random draws, do the following:
  - Only the first point is picked randomly from the scene
  - The second and third point are drawn from the neighborhood of the first point, using a ball of radius *a* (e.g. *a* = 10 cm).



Optimization of the number of draws if object properties / data are known

- Estimation of the complexity (simplified):
  - Probability for “first point on a plane”: 50% (unchanged)
  - Assume: with probability 40%, the first point is not along the border of the patch (i.e. the complete *a*-neighborhood is inside), then two more points drawn from this environment will lead to a good candidate
  - → $b = 0.5 * 0.4 = 0.2$
  - → **$k = 21$**

- Huge difference! **Factor ~700**.

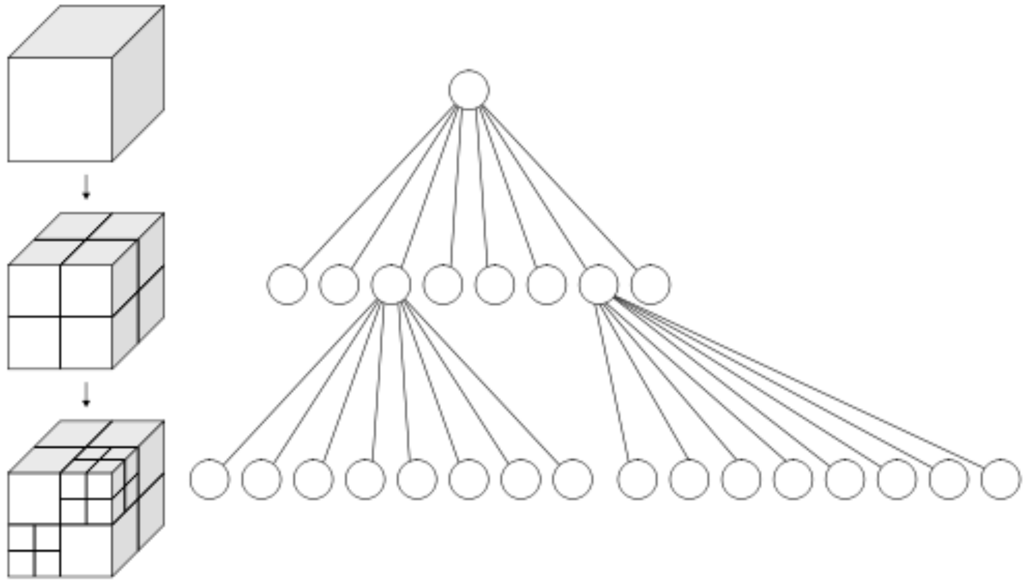


Optimization if object properties are unknown

- How can one proceed if there are very small as well as very large objects in the scene?
  - E.g. in a “street scene”, find facades, stairheads, window sills,…
- How can one proceed
  - If the scale is unknown?
  - If the density of the data is very inhomogeneous (e.g. a high density close to the scanner, a low density at a larger distance)?
- Here: Approach of Schnabel, Wahl, Klein [2007]

Localized sampling

- Idea:
  - Fixed *a* (radius) is not possible, thus do the following:
  - First select the "scale" *a*, then draw the samples
- Use a hierarchic spatial data structure
  - Here: octree (all points are sorted into cells)
  - The "scale" corresponds to one level of the octree



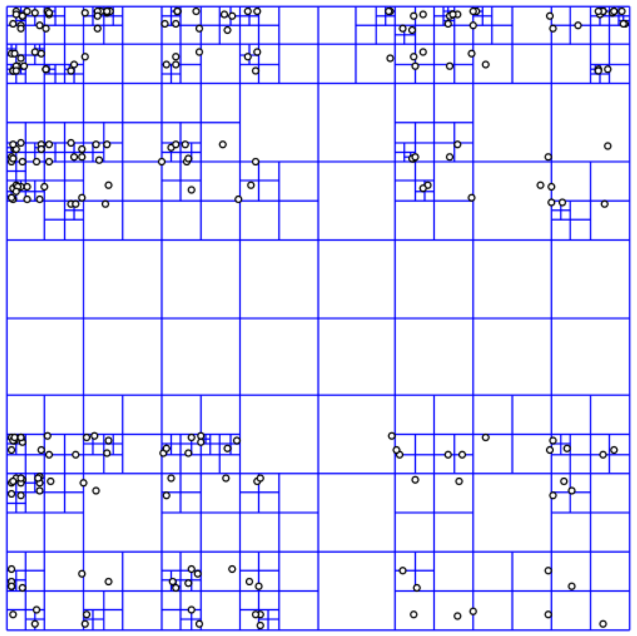


Image source: Wikipedia

Localized sampling

- Algorithm (for 3 points):
  - Draw a point p_1
  - Randomly draw a level of the octree, and pick the cell Z, such that the point p_1 lies in cell Z
  - Randomly draw two more points p_2, p_3 from the same cell Z
- The random selection of the level (and thus, the cell size) replaces the fixed radius a from the previous approach
- Reasoning behind this approach:
  - If a *suitable* cell size is picked, then the probability that p_2, p_3 are on the same object (e.g. planar patch), is relatively high
  - Usually smaller cells will have this property, too.

Localized sampling

p_2, p_3
are drawn subsequently from one of those cells

One of those levels is (randomly) picked

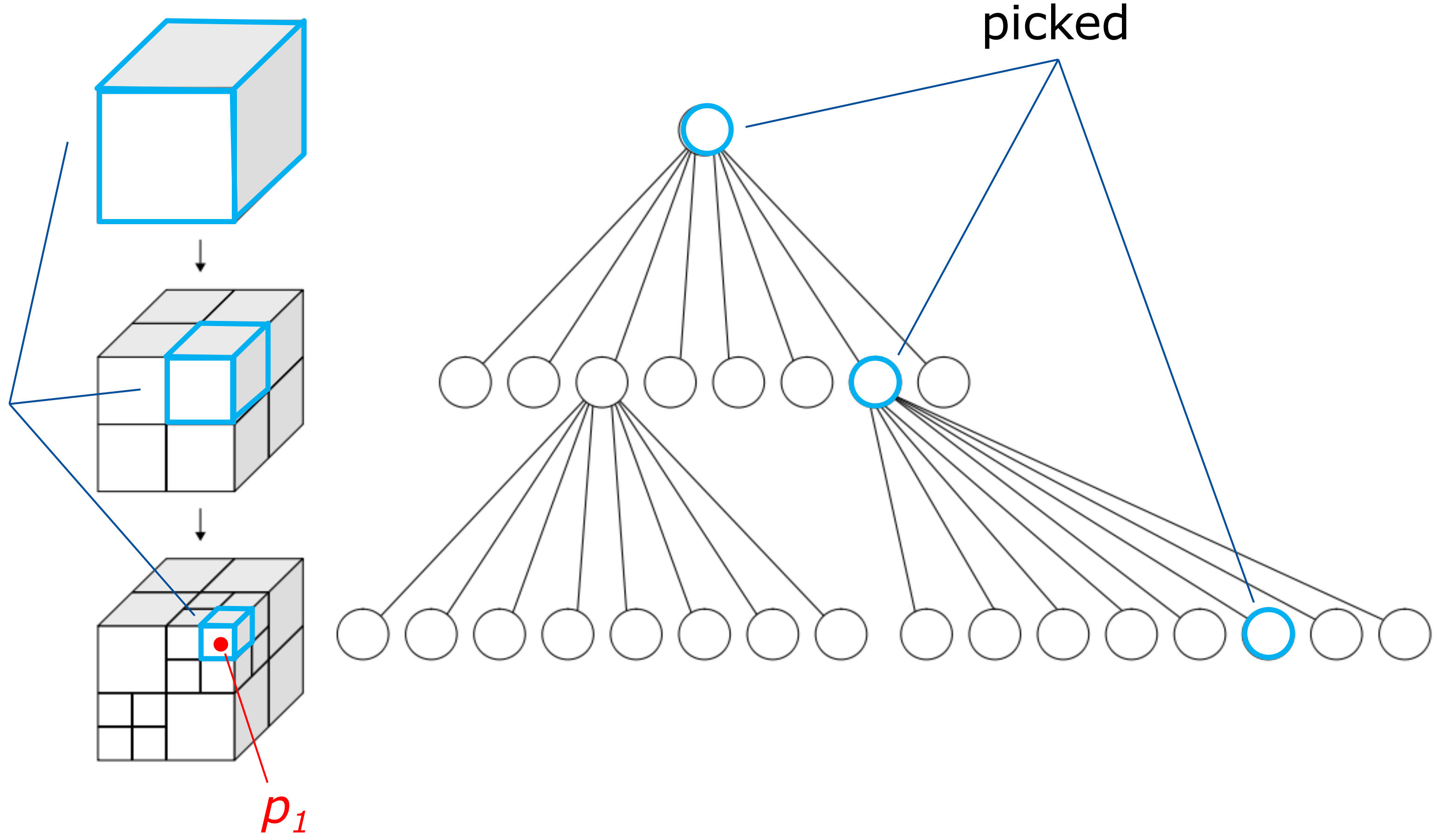
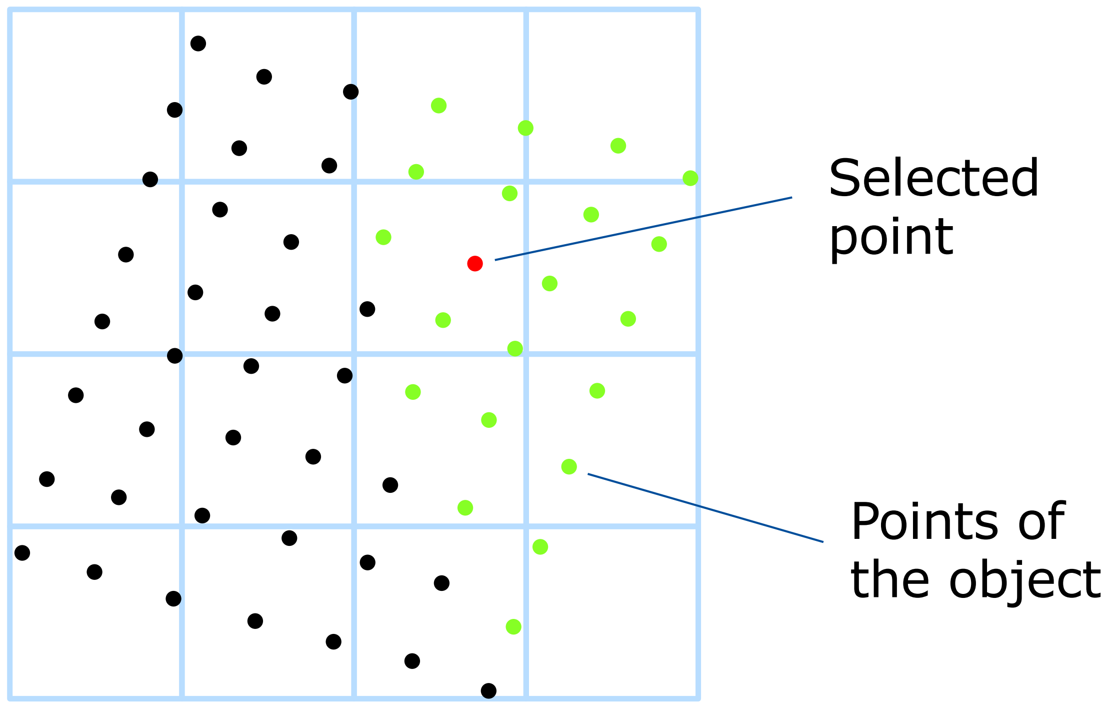


Image source Octree: Wikipedia

Localized sampling - analysis

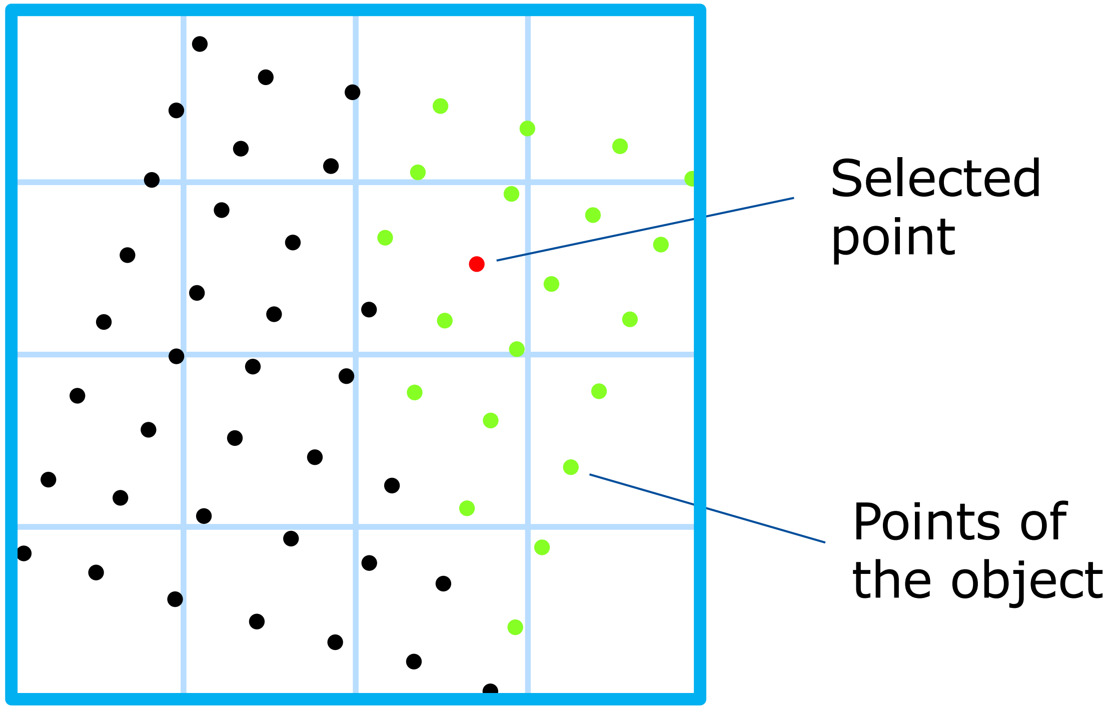
- Selecting the first point p_1 will hit object an object with probability w (same as previously)
- Then, the level of the octree is randomly chosen. Assume the octree has d levels, then the probability for a *suitable* cell is $1/d$
  - This estimate is conservative, since normally, for one suitable cell, all the cells on the way to the leaf are suitable as well
- Assume, that in the picked *suitable* cell, the probability to pick another point of the object is at least 50%
  - I.e., we assume that the cell is sufficiently small, so that at least half of the points in the cell belong to the same object as p_1.

Localized sampling - analysis



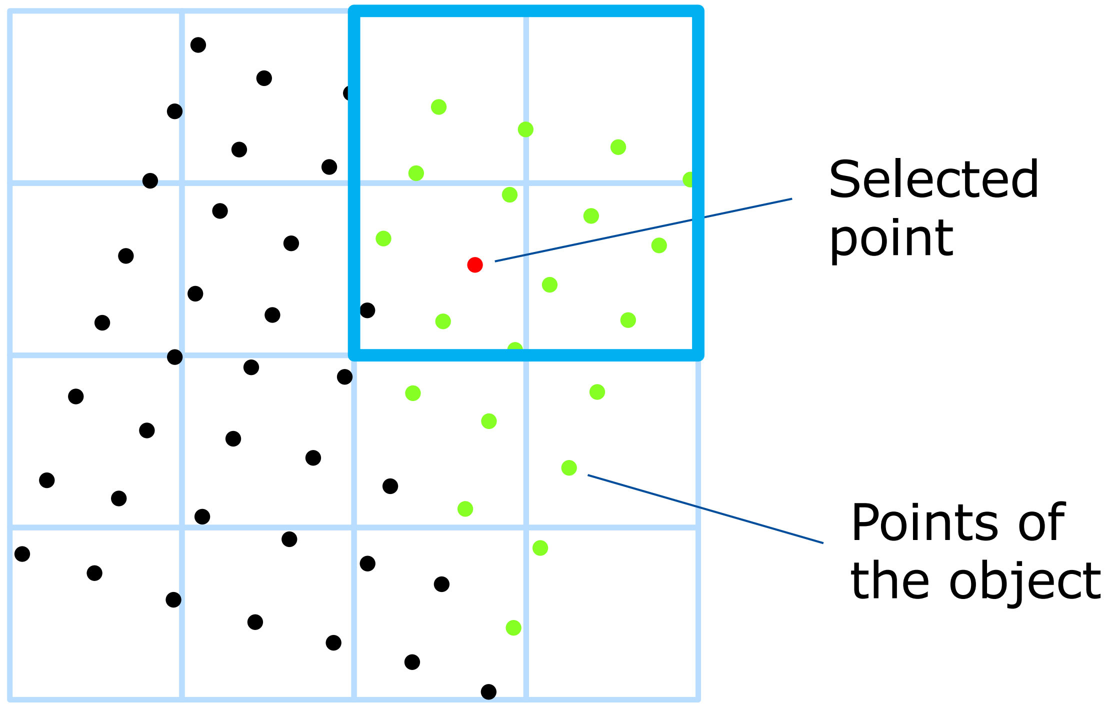
Localized sampling - analysis

< 50%
at level 0



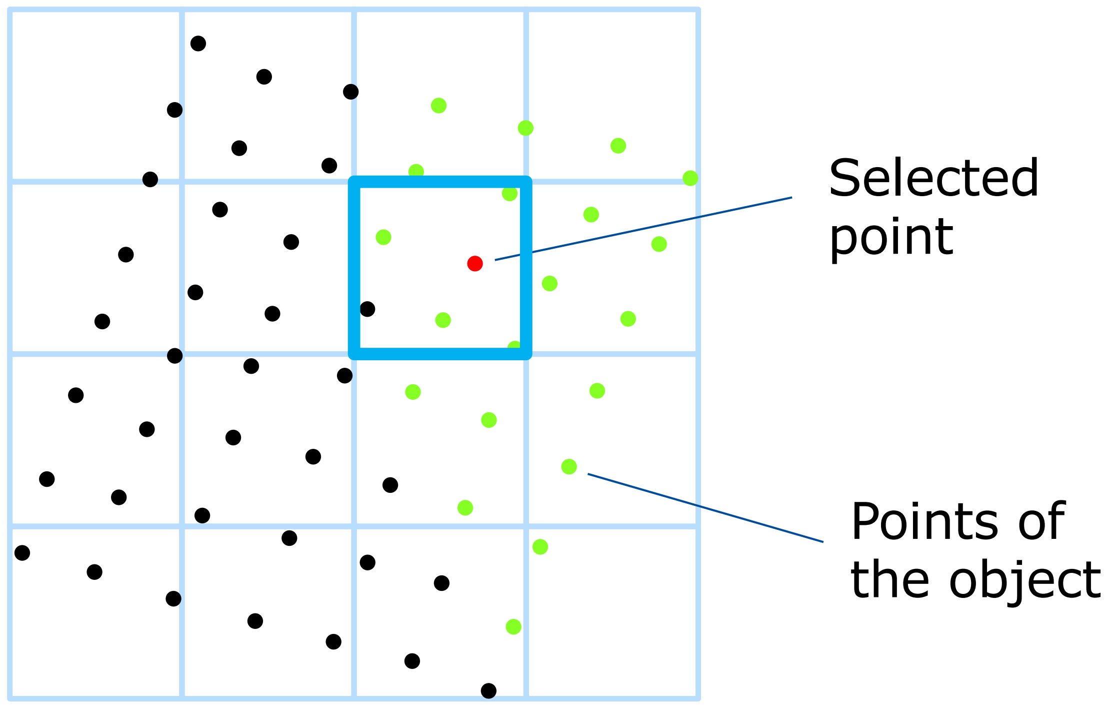
Localized sampling - analysis

> 50%
at level 1



Localized sampling - analysis

> 50%
at level 2



Localized sampling - analysis

- Probability for picking the first point p_1: w
- Probability for selecting a suitable level of the octree: $1/d$
- Selection of the other points $p_2 \ldots p_n$ (assuming a probability for the object of 50% = 1/2): $(1/2)^{n-1}$
- Overall, this results in:

$$b = w \cdot \frac{1}{d} \cdot \left(\frac{1}{2}\right)^{n-1}$$

Localized sampling - analysis

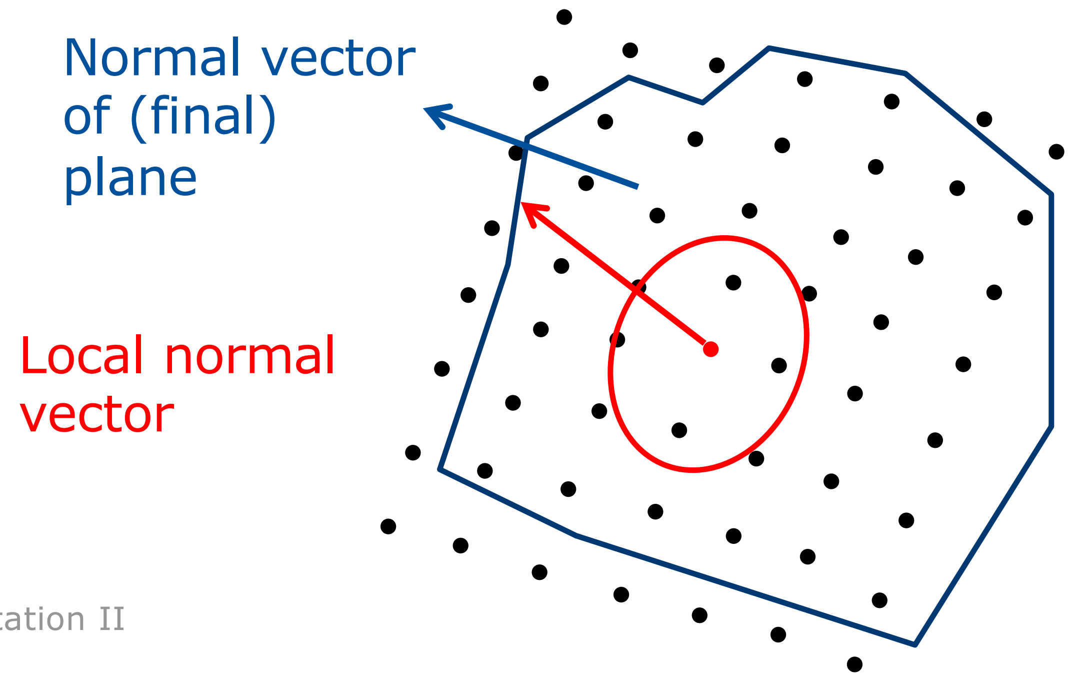
- For the example from above
- Assume the octree has a total of 5 levels
  - Assuming 1 M points, this means a cell (in a leaf) has on average 1000000 / 8^5 = 30,5 points
- 50% probability that the first point is selected on a segment
- $1/d = 1/5$
- $n=3$ (points required to define a plane)
- → $b = 0.5 \cdot \frac{1}{5} \cdot \left(\frac{1}{2}\right)^2 = \frac{1}{40}$
- → $k = 182$
- Improvement by a factor of: 14735 / 182 = **81**

Localized sampling - analysis

- This factor increases with the number of elements in the scene
  - E.g.: 50% no segments, remaining points are on **100** segments
  - Simple method (increased): k = 368412
  - Localized sampling (unchanged): k = 182
  - → Factor: 2024 (…one hour vs. 1.8 seconds)
- This can be further improved if the level of the octree is not selected uniformly, but proportional to the size of the resulting *consensus set*.
  - Initialize the probability for the selection of level i with $1/d$ (uniformly)
  - Increase the probability for level i, if the selection of this level leads to larger *consensus sets* (sum of scores)
  - For details, see equation (7) in [Schnabel, Wahl, Klein, 2007].

Further remarks

- The number of draws can also be reduced by using more information
  - E.g. if local normal vectors are available, then (point and local normal vector) is sufficient to define a plane.
    In this case, only one point needs to be drawn instead of 3.
    Therefore, $b=w$ instead of $b=w^3$
  - This will only work if the normal vector of each point can be determined with sufficient accuracy.



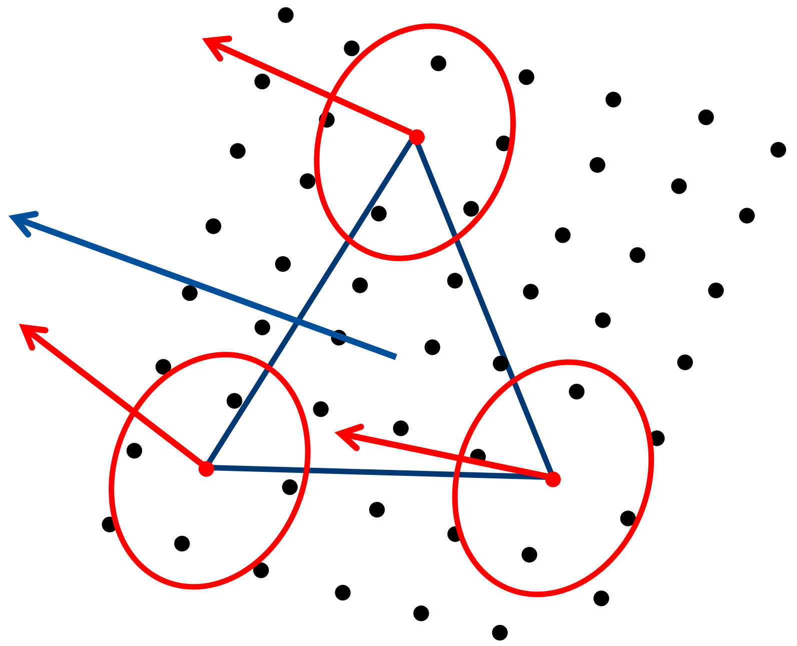
RANSAC: Optimization 2: Evaluation of draws

Evaluation of draws

- Reminder: for small b the cost is: $O(C/b)$
  - So far, we tried to optimize *b*
- Now: optimize cost of evaluation, *C*
- So far, we used the following algorithm:
  - For each draw:
    - Loop over all points. For each point, test if it fits to the model
    - If yes → increase count (score)
  - The draw which leads to the highest count (the largest *consensus set*) wins
- Costly, since the scenes may have millions to billions of points.

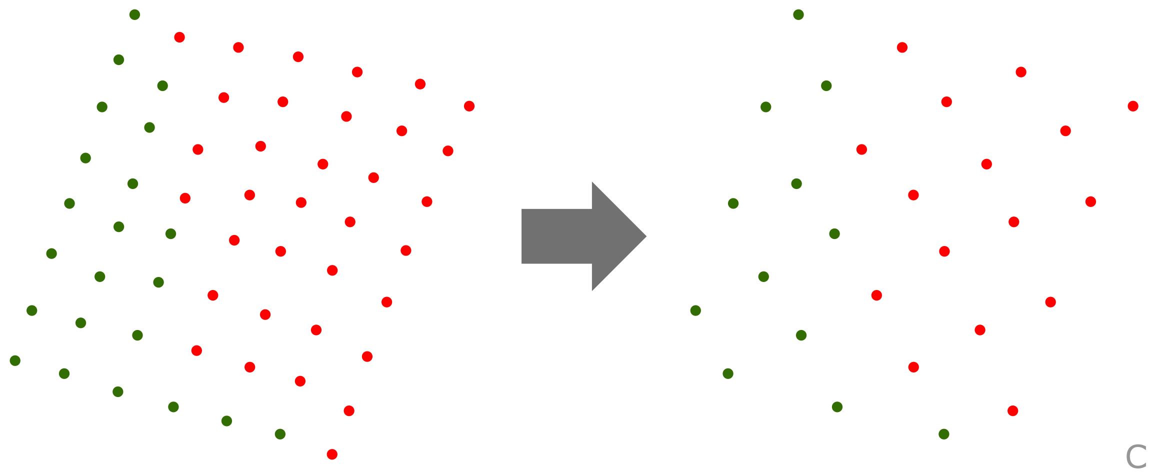
Evaluation of draws

- One possible heuristic: only compute the score, if it seems to be promising
  - E.g.: draw 3 points for a plane. Compute the score only if the local normal vectors are compatible with the normal vector defined by the plane through the three points (compatibility given by a threshold for the maximum angle deviation).



Evaluation of draws

- General reasoning:
  - Often, the objects to be segmented contain a large number of points
  - E.g. 10^3 … 10^6 points in one planar segment
  - But for the RANSAC algorithm, all it needs to know is which draw leads to the largest score. I.e., the relative ranking among the random draws is important, not their absolute score
- Consequence:
  - To find the (relative) ranking, a subset of points is sufficient.



Evaluation of draws

- The set of points P is partitioned into *r* subsets (using random selection)

$$P = P_1 \cup P_2 \cup \ldots \cup P_r \quad \text{with } P_i \cap P_j = \varnothing \text{ for } i \neq j$$

- Example: 1 M points are partitioned into 10 subsets of 100,000 points each
- Note: forming the subsets is “sampling without replacement”
  - Hypergeometric distribution
- We compute the score first on the subset P_1

Hypergeometric distribution

- Let
  - Population size: *N*
  - Probability of success: *m/N*
  - Number of draws: *M*
  - Number of observed successes: *i*
- Then, the hypergeometric distribution is given by the probability for *i* successes:

$$P(X=i)=\frac{\binom{m}{i}\binom{N-m}{M-i}}{\binom{N}{M}}$$

Hypergeometric distribution

- Typical example:
  - There are *N* marbles in an urn
  - Of those, *m* are white, the remaining ones are black
  - *M* marbles are drawn without replacement
  - Then, the probability that *i* white marbles are drawn is given by the formula (of the previous slide):

Possibilities to draw *i* white marbles from the set of *m*

Possibilities to draw the remaining *M*-*i* marbles from the remaining *N*-*m* black marbles.

$$P(X=i)=\frac{\binom{m}{i}\binom{N-m}{M-i}}{\binom{N}{M}}$$

Total possibilities to draw M marbles from a total of N.

Hypergeometric distribution

- Properties of the hypergeometric distribution:
- Expectation

$$M \cdot \frac{m}{N}$$

- Variance

$$M \frac{m}{N} \frac{N-m}{N} \frac{N-M}{N-1}$$

- Now define the following interval, expectation +/- standard deviation:

$$f(N,M,m) = \frac{Mm \pm \sqrt{\frac{Mm(N-m)(N-M)}{N-1}}}{N} = [a,b]$$

Evaluation of draws

- Now apply this to our problem…
- We have computed the score for the subset P_1 of points and want to compute the "expected" score for the total set P
- I.e., from a random sample, we want to infer the situation for the overall set
- Induction formula (statistics):

$$m = -1 - f(-2 - M, -2 - N, -1 - i)$$

- Note this will yield an interval (since f(…) yields an interval).

Evaluation of draws

- From that, the following algorithm can be derived:
  - First, compute the score s only for the subset P_1
  - Extrapolate from P_1 to P:

$$m = -1 - f(-2 - |P_1|, -2 - |P|, -1 - s)$$

- To rank the draws:
  - Assume: two draws with score s and s'
  - For both, extrapolate (yields intervals in both cases):

$$m = -1 - f(-2 - |P_1|, -2 - |P|, -1 - s_1) = [a, b]$$
$$m' = -1 - f(-2 - |P_1|, -2 - |P|, -1 - s_1') = [a', b']$$

Evaluation of draws

- If the two intervals do not overlap, select the draw with the larger values to be the winner
- If they overlap, the winner cannot be chosen. Enlarge the set by adding the points from P_2:

$$m = -1 - f\left(-2 - (|P_1| + |P_2|), -2 - |P|, -1 - (s_1 + s_2)\right) = [a, b]$$

$$m' = -1 - f\left(-2 - (|P_1| + |P_2|), -2 - |P|, -1 - (s_1' + s_2')\right) = [a', b']$$

- If the two intervals now do not overlap anymore, select the draw with the larger values as the winner
- Otherwise, enlarge the set again to $P_1 \cup P_2 \cup P_3$
- → This yields an algorithm where the point set which is used to compute the score is selected "only as large as necessary".

Evaluation of draws

- Example:
  - Total set *P* of 1000 points
  - Partitioned into $P_1,\dots,P_{10}$ with 100 points each
- RANSAC
  - Assume, one draw leads to a consensus set (subset of P_1) of 30
  - Assume, another draw leads to a consensus set (also a subset of P_1) of 40
  - Extrapolation yields the intervals:
    - For 30: [260, 347]
    - For 40: [356, 448]
  - The intervals do not overlap, thus select the second draw as the winner (without touching any further points).

Evaluation of draws

- Example 2:
  - Now, assume the scores are 32 and 40, then the intervals are:
  - For 32: [279, 367]
  - For 40: [356, 448]
- The intervals overlap. Therefore, count the consensus set for the set $P_1 \cup P_2$
  - Assume, in P_2 there is the same score. Then, the total set is 1000, the subset is 200, and the scores are 32+32 = 64 and 40 + 40 = 80
  - For this, the following intervals are extrapolated:
    - For 64: [292, 351]
    - For 80: [370, 432]
- The intervals do not overlap anymore, the second draw is the winner.

Evaluation of draws

- The larger the set which is used to obtain the consensus set, the smaller the uncertainty (the width of the interval)
- Therefore, with increasing addition of points, the determination of the winner is possible
- Example. Simulation with score = 30% of the respective subset

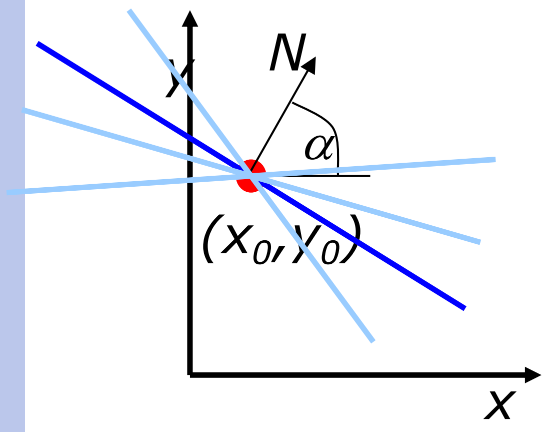
Total.	Subset.	Score	Interval	Width of interval
1000	100	30	[260, 347]	86
1000	200	60	[273, 330]	58
1000	300	90	[279, 323]	44
1000	400	120	[283, 318]	35
1000	500	150	[286, 315]	29
1000	600	180	[288, 312]	24
1000	700	210	[291, 310]	19
1000	800	240	[293, 307]	14
1000	900	270	[295, 305]	10
1000	1000	300	[300, 300]	0

Other remarks

- RANSAC is a general method for robust parameter estimation (although we (mis-) used it here in the special context of segmentation)
  - E.g. for “spatial resection” (this is an example in the original paper of Fischler und Bolles 1981).

Segmentation: The Hough Transformation

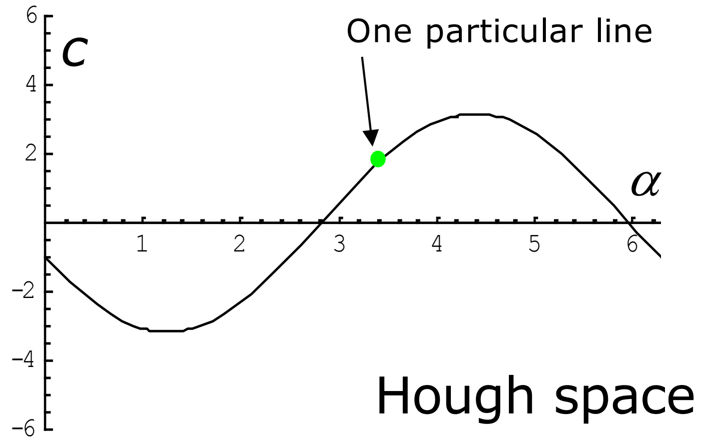
The Hough Transformation (PVC Hough, 1962)

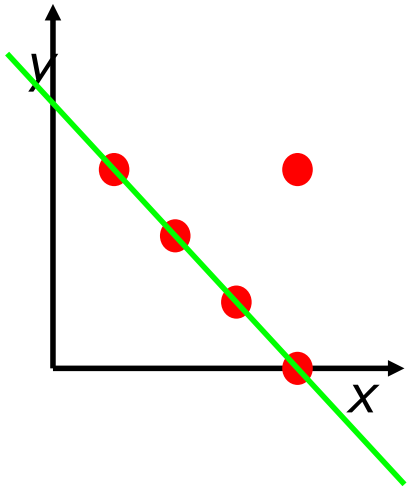


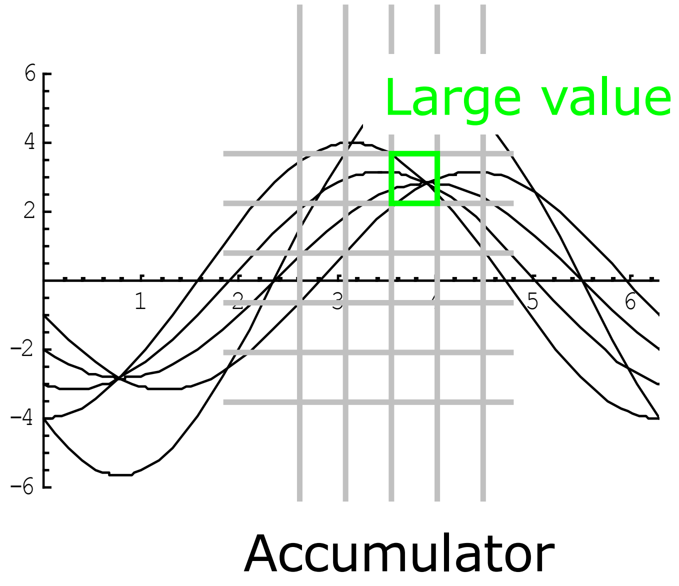
$$N = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}$$

$$x\cos\alpha + y\sin\alpha + c = 0$$

$$c = -(x_0 \cos\alpha + y_0 \sin\alpha) = c(\alpha)$$

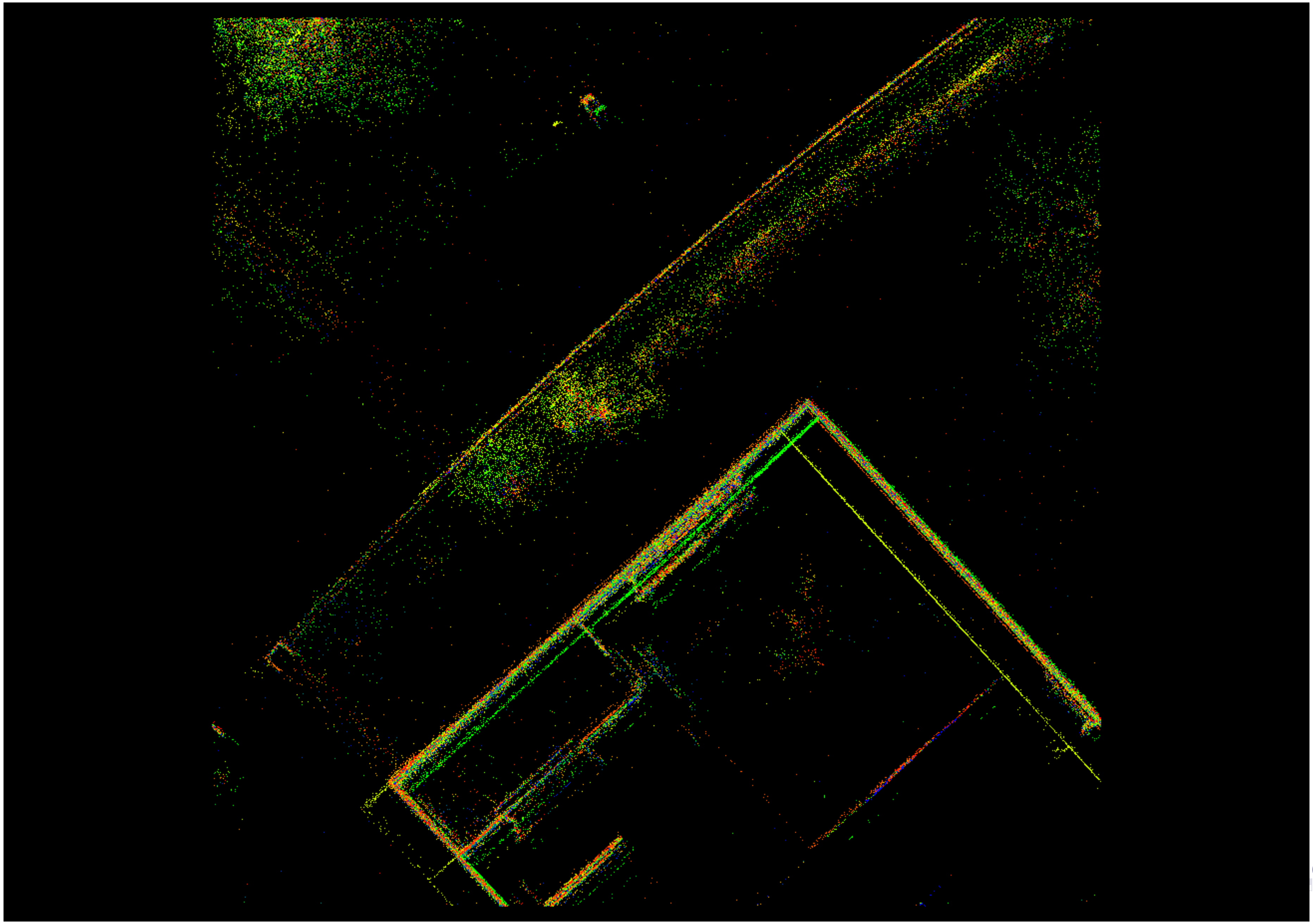




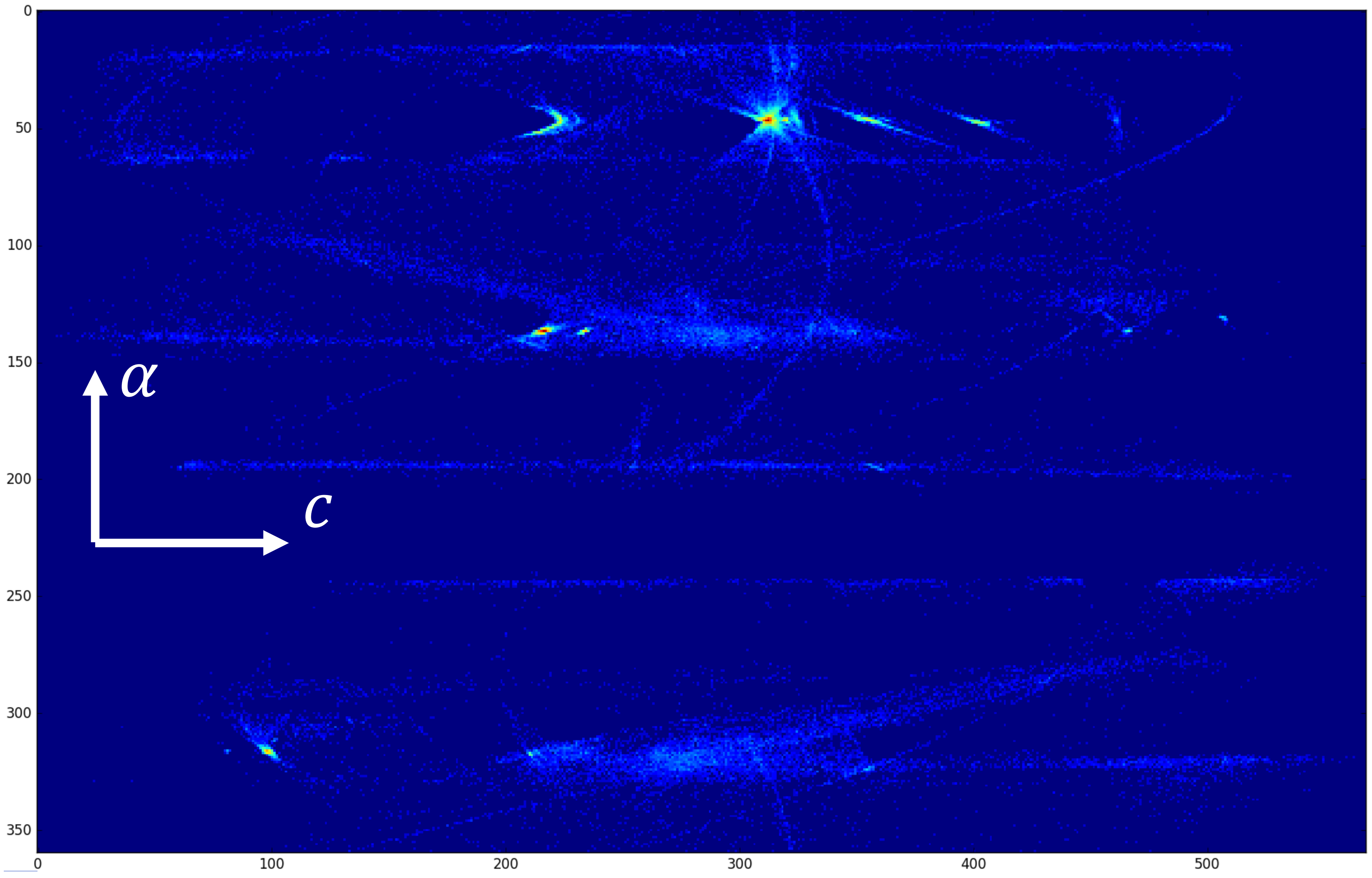


(Note: a point with known normal vector corresponds to a single point in Hough space)

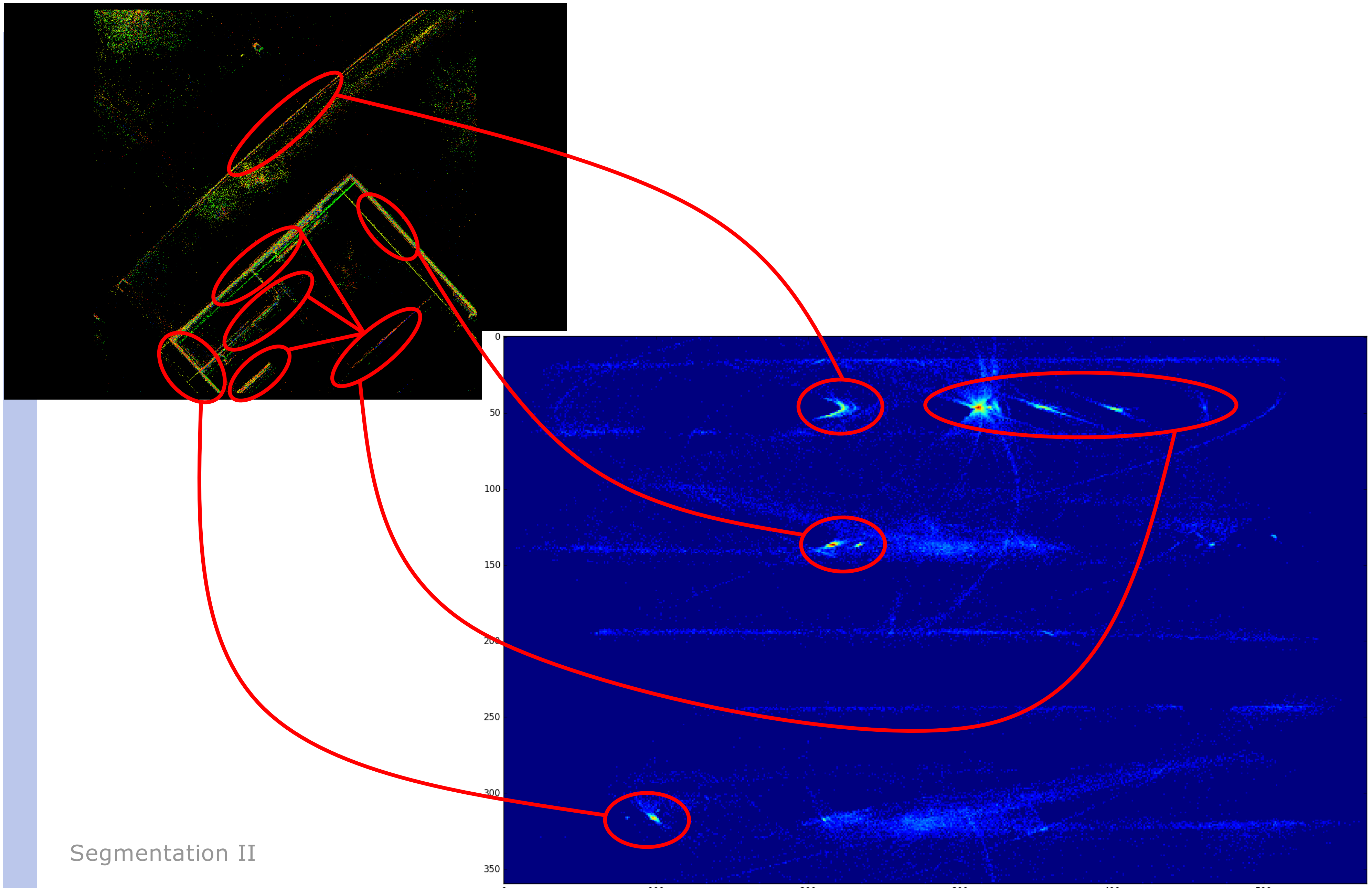
Example: Mobile Mapping Scans (topview, horizontal points removed, one color per scan)



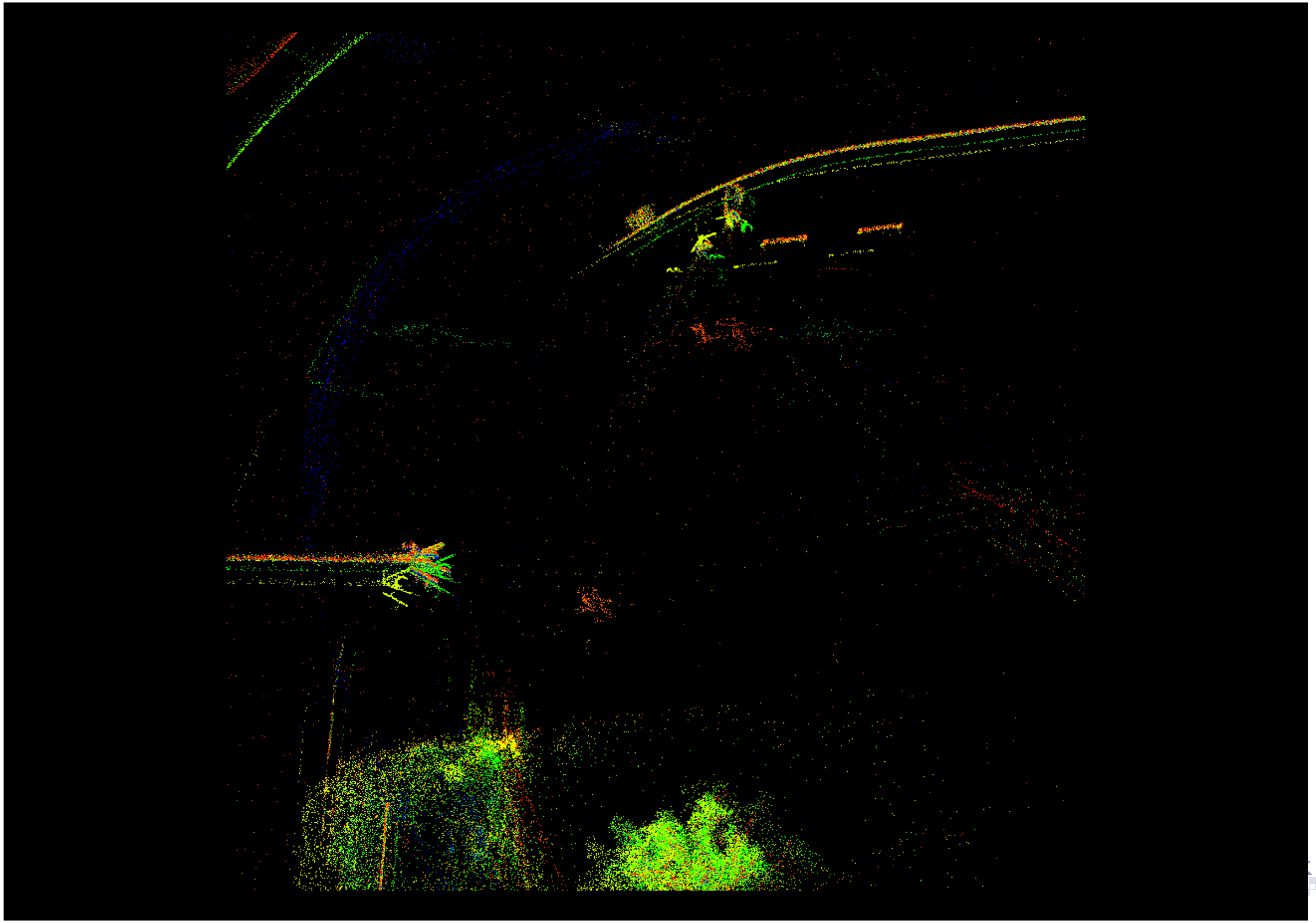
2D Hough space



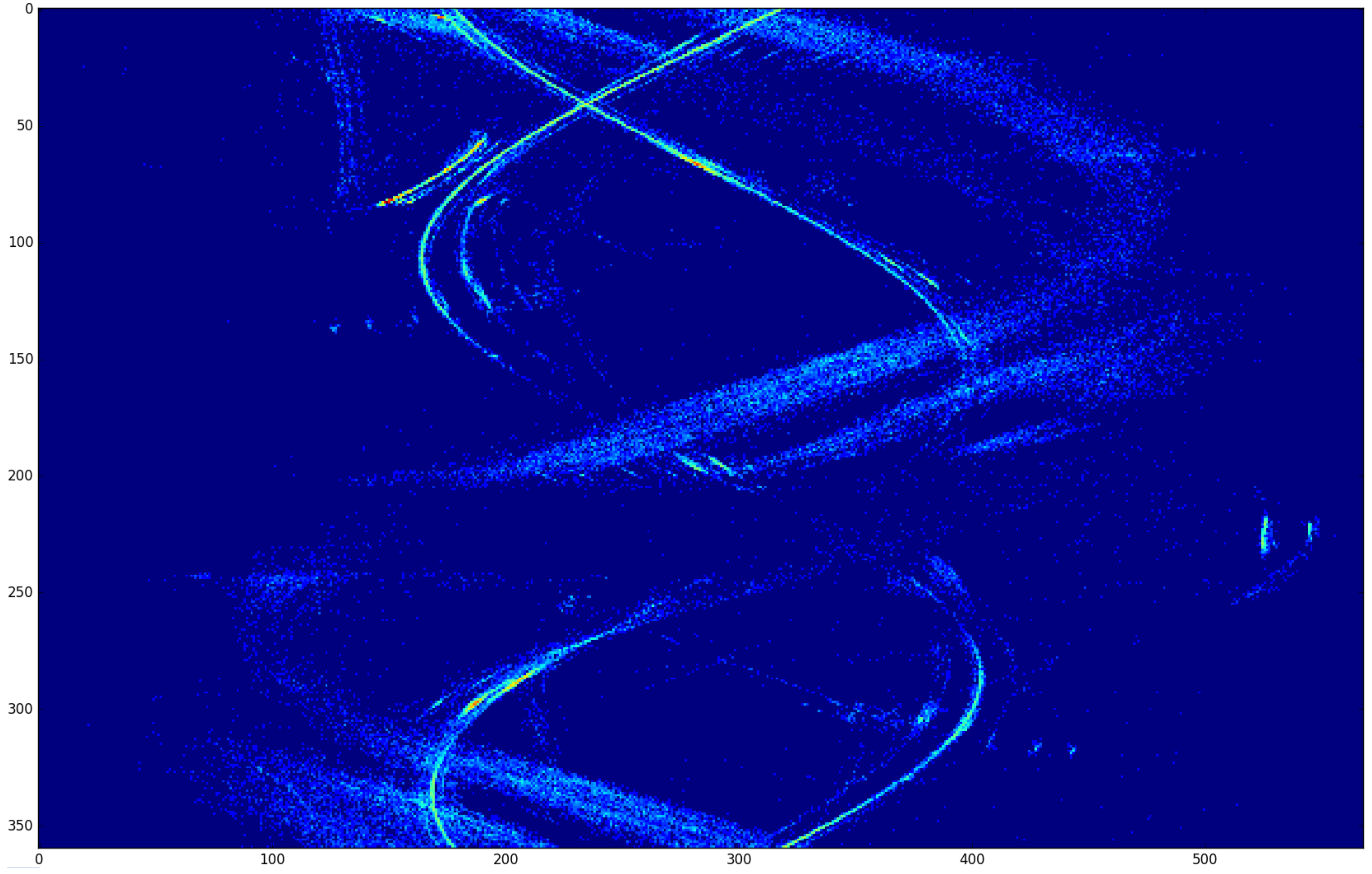
Correspondences



Example: Mobile Mapping Scans (topview, horizontal points removed, one color per scan)



2D Hough space: no planes, two groups of poles (traffic lights) lead to sine shaped curves



The Hough transform for planes

- Plane equation:

$$ax + by + cz + d = 0$$

- Similar to the case of a line: parameterization of the normal vector in terms of the (now: *two*) angles (standard formula for polar coordinates):

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\alpha\cos\beta \\ \sin\beta \end{bmatrix}$$

- Then, the Hough space has three dimensions:

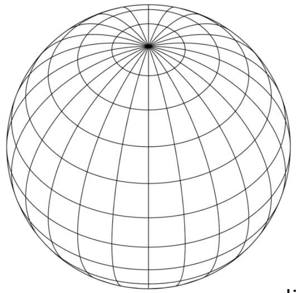
$$\alpha, \beta, d$$

Algorithm

- (Often: normalize the coordinates of the point cloud first)
- Define the discretization of the Hough space, i.e. the size of the accumulator cells in α, β, d
- Loop over all points:
  - Each single point of the point cloud defines a surface in Hough space, i.e. requires a loop over α, β, d and increment of all corresponding cells
- Find the maximum in Hough (accumulator) space
- Find all points in the original point cloud which have contributed to this maximum. Form one (or more) regions with minimum required size
- (Optionally) re-estimate the plane parameters of those regions, e.g. by using least squares adjustment
- Remove the point count from the accumulator and go back to the determination of the maximum

Hough Transformation: Problems

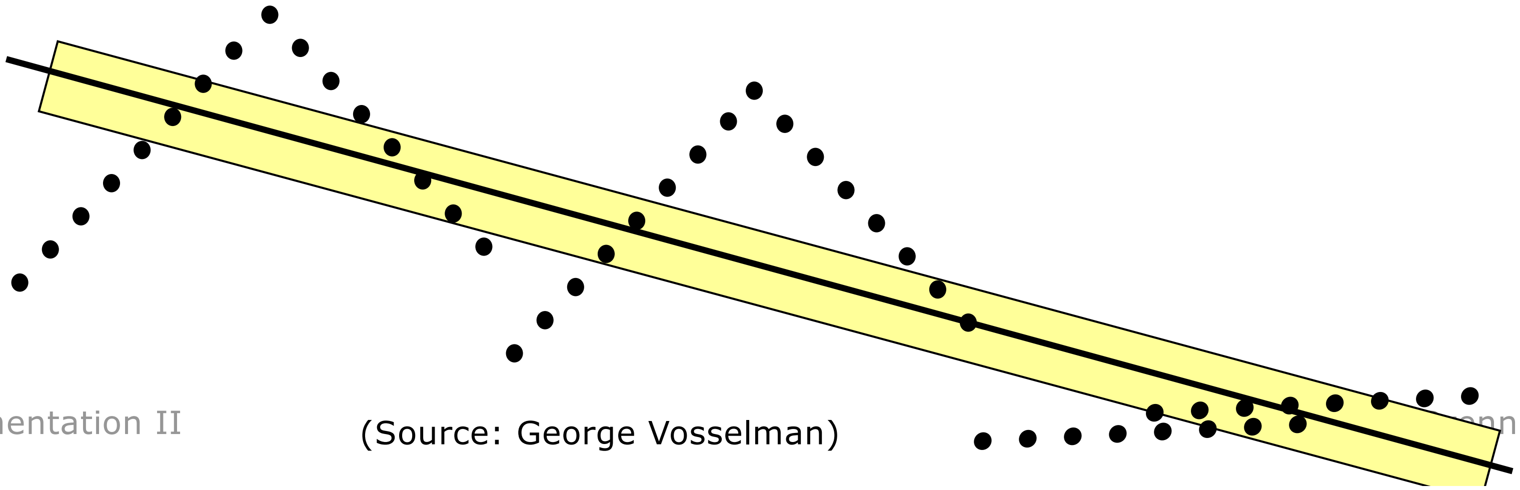
- Hough space is discretized
  - Selection of the correct size can be tricky
    - Too large → parameters are not well determined
    - Too small → maxima are distributed across many cells
  - In any case, maximum may be spread over 2 cells
- If the algorithm is implemented as described here, note that:
  - Discretizing with constant increments for α and β induces different cell sizes (the cell size decreases with increasing latitude). This would prefer larger cells, i.e. planes with normal vectors in the XY plane.


Source: media.istockphoto.com

    - Solution 1: Normalize each accumulator counter by its area on the Gaussian sphere
    - Solution 2: Select the increments in α dependent on β (increments of longitude depending on latitude).

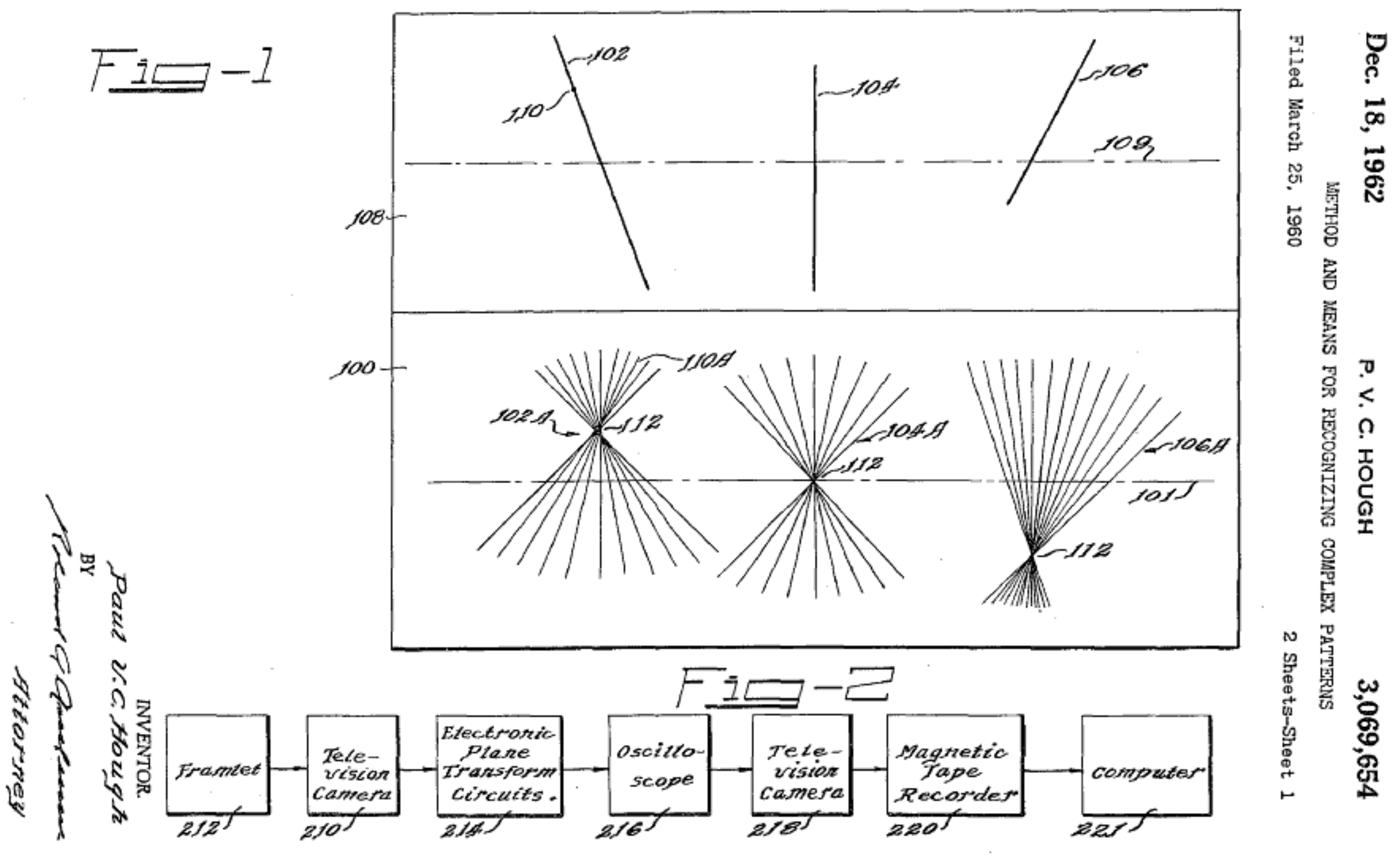
Hough Transformation

- Can be used to find arbitrary, parameterized objects: lines, circles, planes, cylinders…
- Higher dimensions get intractable (space (memory), time)
- Sometimes, one Hough transform can be split into two transforms of lower dimension
  - E.g. cylinder: 5 parameters. Instead of 5 dimensional Hough space: first determine axis (2D), then point and radius (3D)
- The Hough transform is computed globally. This can be problematic, e.g.:


(Source: George Vosselman)

References

- Schnabel, Wahl, Klein, Efficient RANSAC for Point-Cloud Shape Detection, Computer Graphics Forum, 26(2), 2007. Blackwell Publishing.
- Fischler, Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, Comm. of the ACM, 24(6), June 1981. [RANSAC]
- Paul V. C. Hough, METHOD AND MEANS FOR RECOGNIZING COMPLEX PATTERNS, US Patent 3069654, filed Mar. 25, 1960, issued Dec. 18, 1962.



Dec. 18, 1962 P. V. C. HOUGH 3,069,654

METHOD AND MEANS FOR RECOGNIZING COMPLEX PATTERNS

Filed March 25, 1960 2 Sheets-Sheet 1