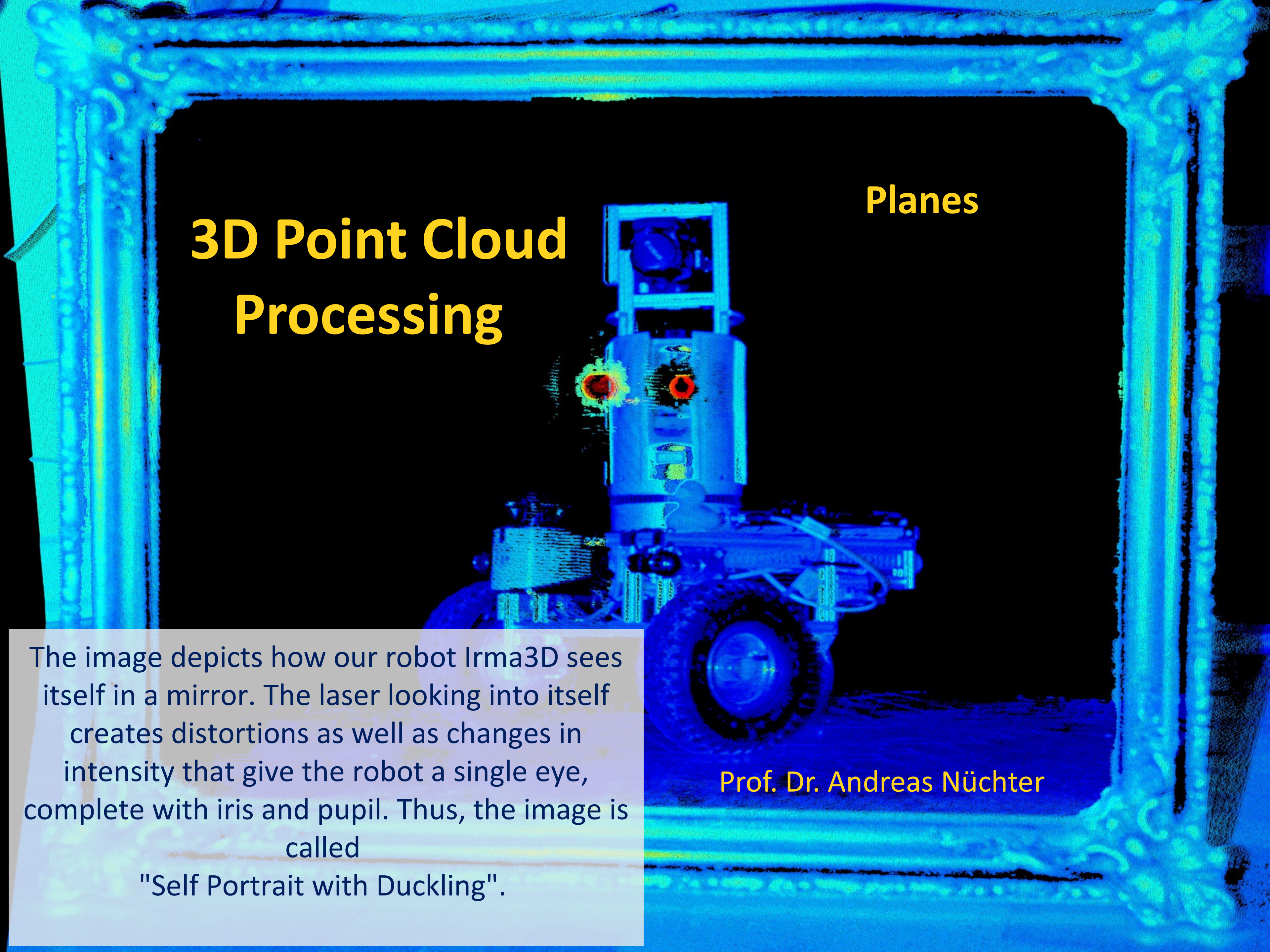


3D Point Cloud Processing

Planes

The image depicts how our robot Irma3D sees itself in a mirror. The laser looking into itself creates distortions as well as changes in intensity that give the robot a single eye, complete with iris and pupil. Thus, the image is called
"Self Portrait with Duckling".

Prof. Dr. Andreas Nüchter

SIFT Example (1)

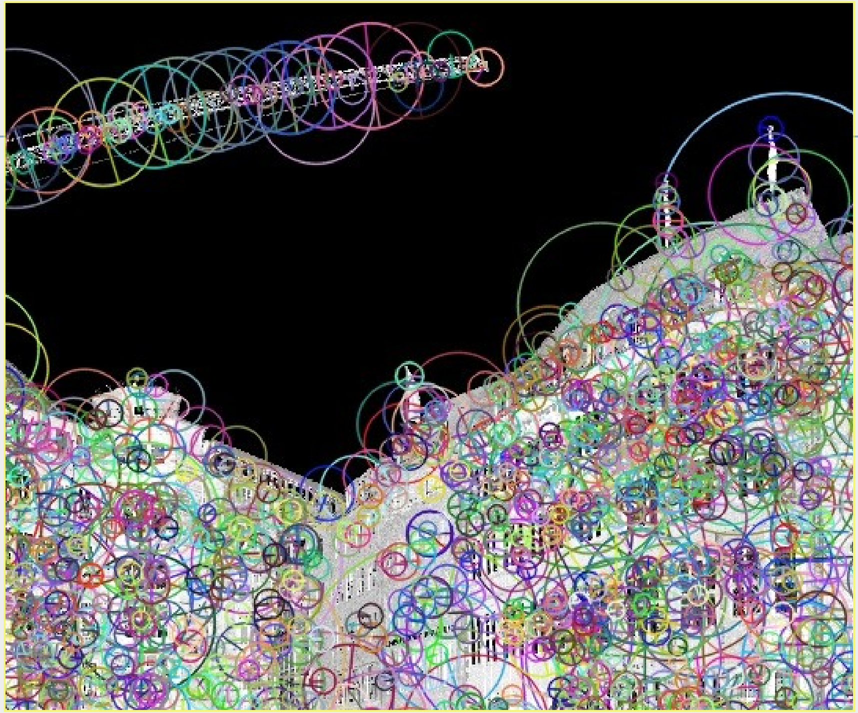
- Typical 30000 features in an image of 3.6 Megapixel

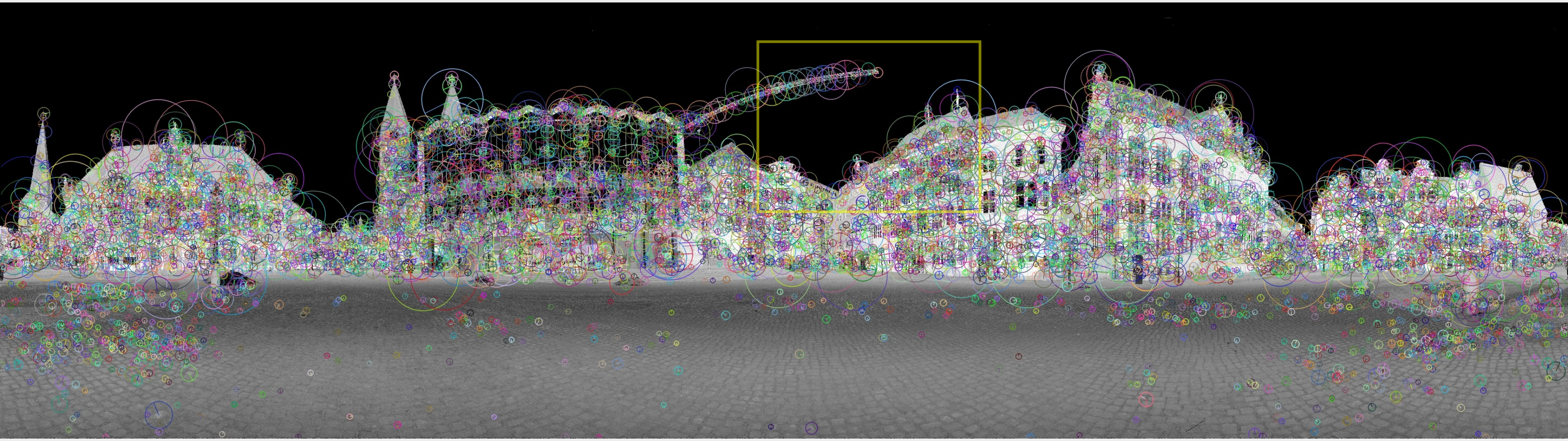
- Example:

SIFT Example (2)

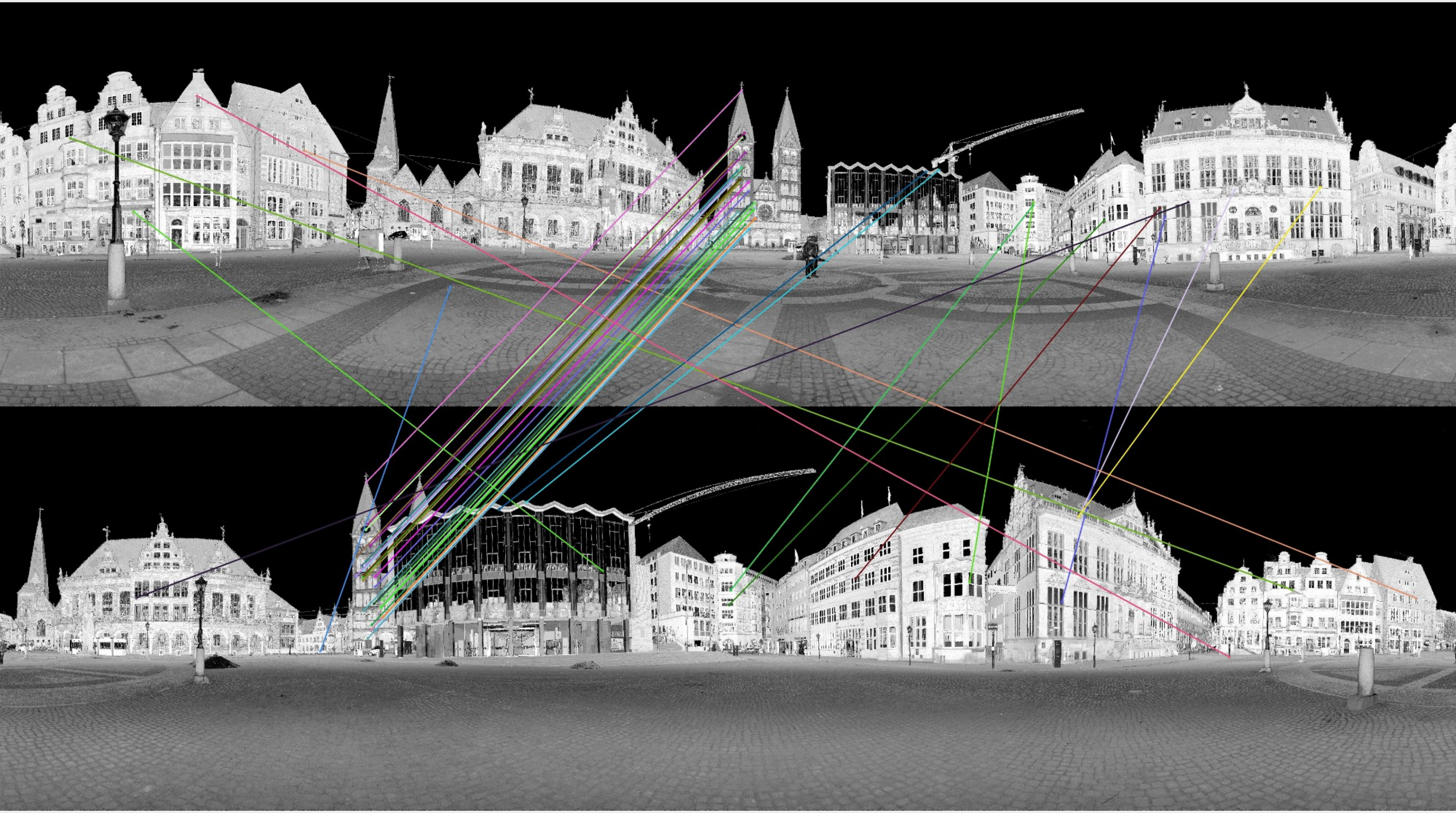
- Typical 30000 features in an image of 3.6 Megapixel

- Example:



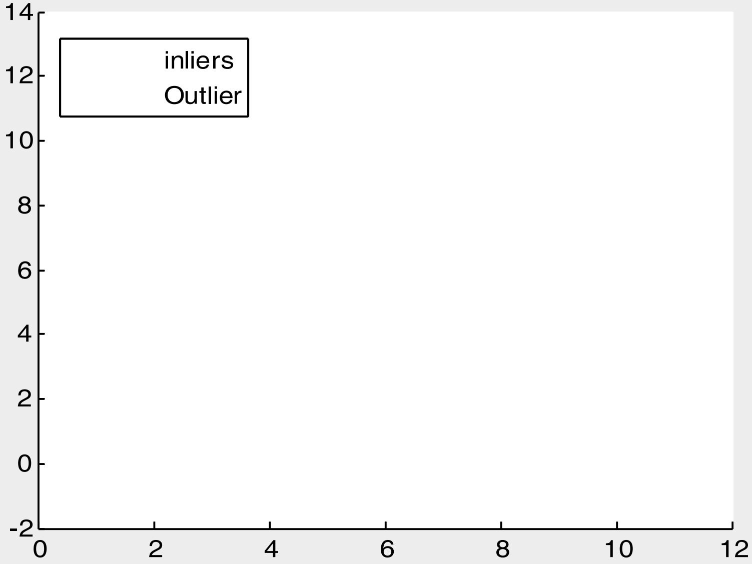


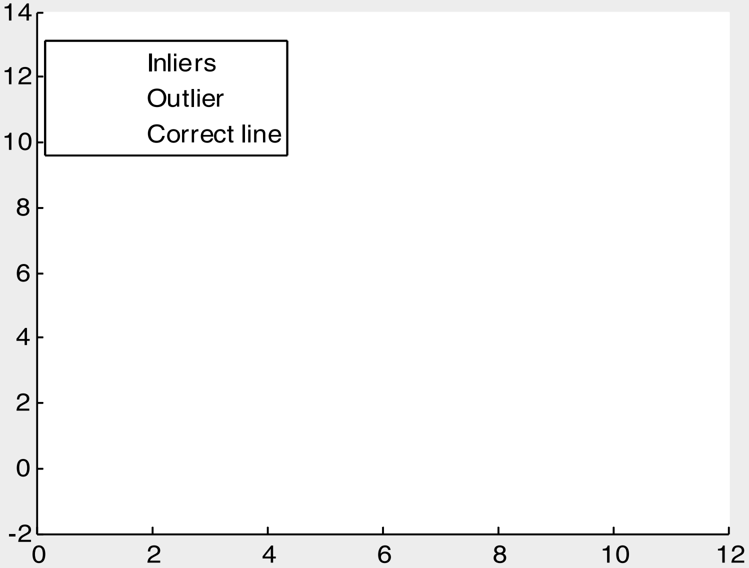
Remember: Feature-Based Registration



Minimization using Least Squares

- This minimization does not tolerate “outliers”





Robust Estimator for Data with Outlier

- The M-estimator and Least-Median-of-Squares (LMedS) estimator cannot cope with 50% outliers

- Solution: The RANSAC (RANdom SAmple Consensus) Algorithm
  - Developed by Fischler and Bolles
  - One of the most importtant techniques in computer viso
  - Can cope with 50% or more outlier

The RANSAC Algorithm – In General

- Generate *M* (a predefined number) of Model hypotheses, from which all a calculated by a minimal set of points
- Evaluate all hypotheses
- Calculate the remaining error using all data.
- Points with errors smaller than a threshold are classified as “Inlier”
- The hypothesis with a maximal number of “Inlier” is chosen. Aterwards the hypotheses is re-estimated, using only the inlier.

The RANSAC Algorithm – Formal

k := 0

Repeat until P(better solutions exists) < η

(cost function C and step counter k)

k := k + 1

I. Hypothesis estimation

(1) Select random set $S_k \subset U, |S_k| = m$

(2) Calculate parameter $p_k = f(S_k)$

II. Verification

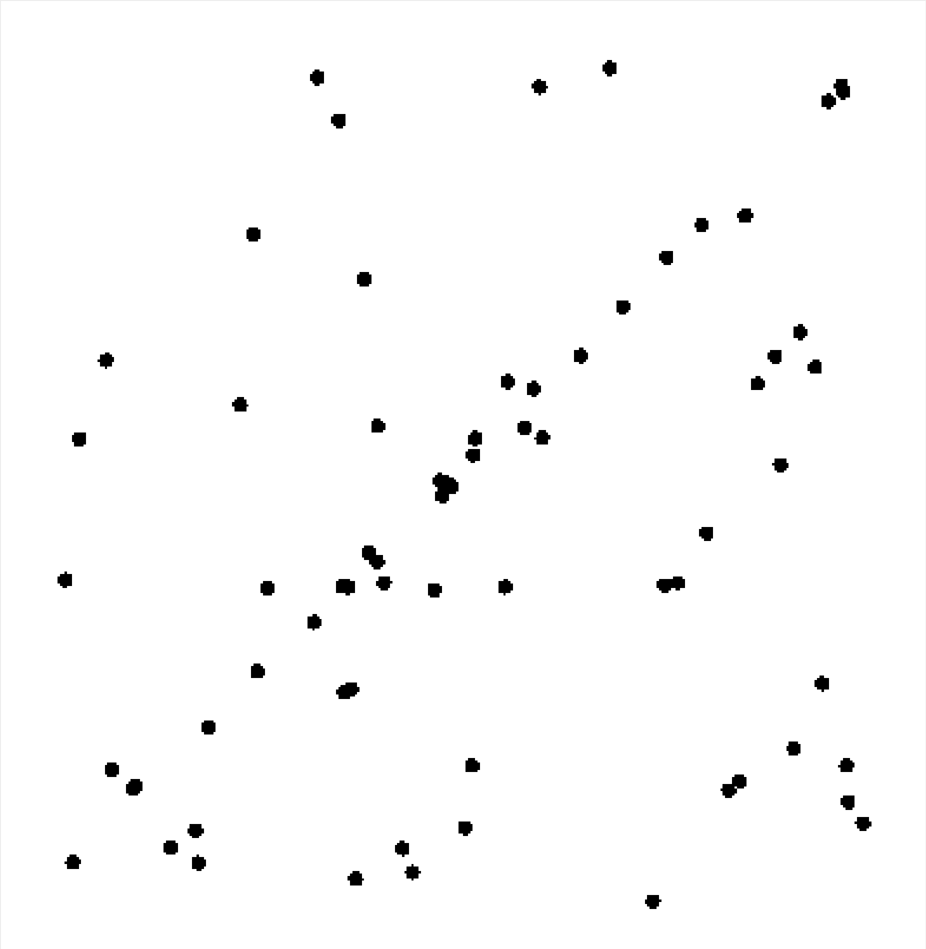
(3) calculate costs $c_k = \sum_{x \in U} \rho(p_k, x)$

(4) if $C^* < C_k$ then $C^* := C_k$, $p := p_k$

end

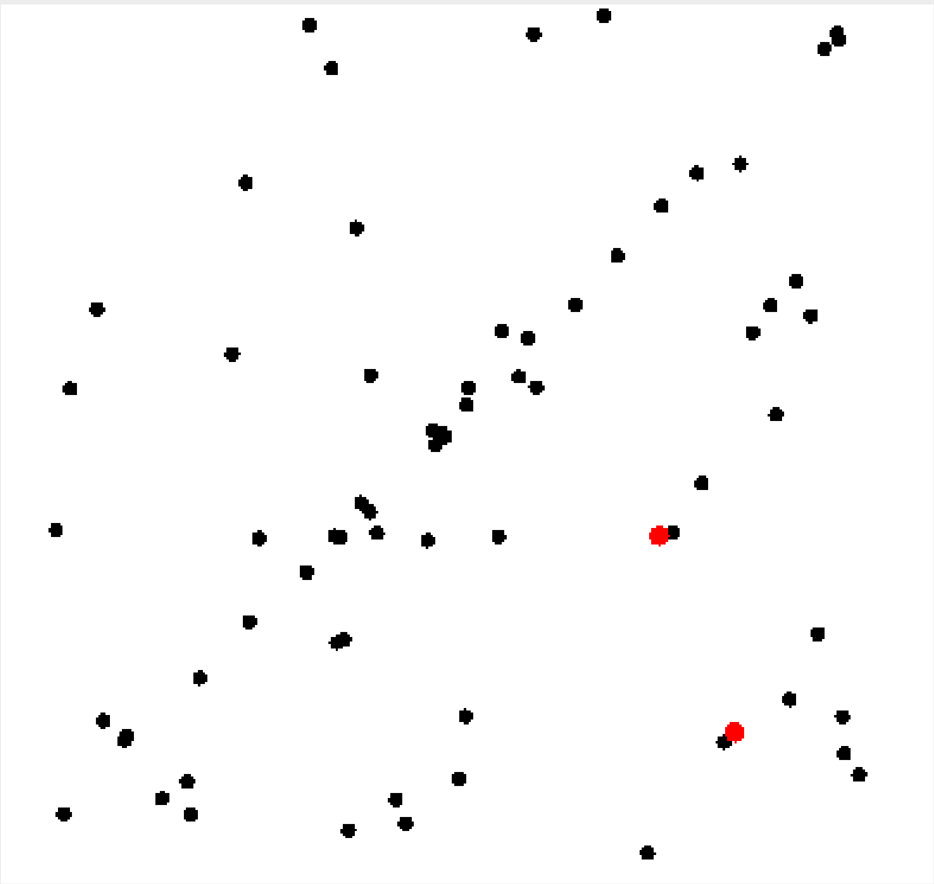
Example: Line Detection with RANSAC (1)

- Given the following set of points



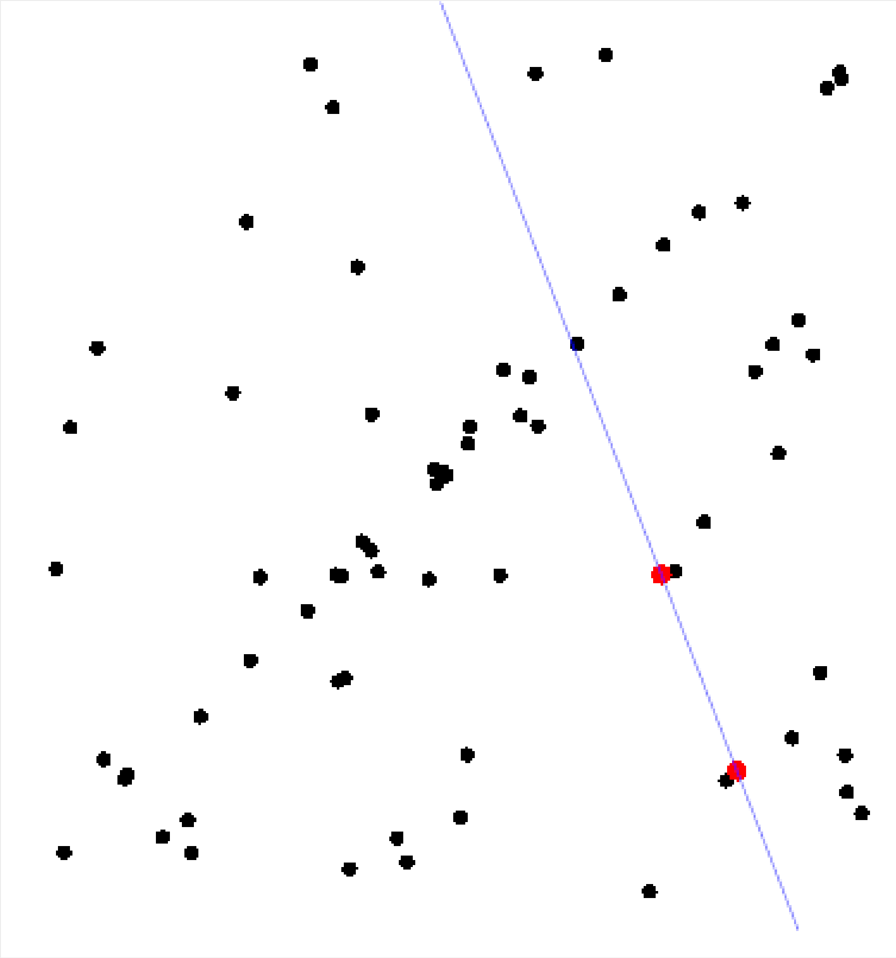
Example: Line Detection with RANSAC (2)

- Select points @ random



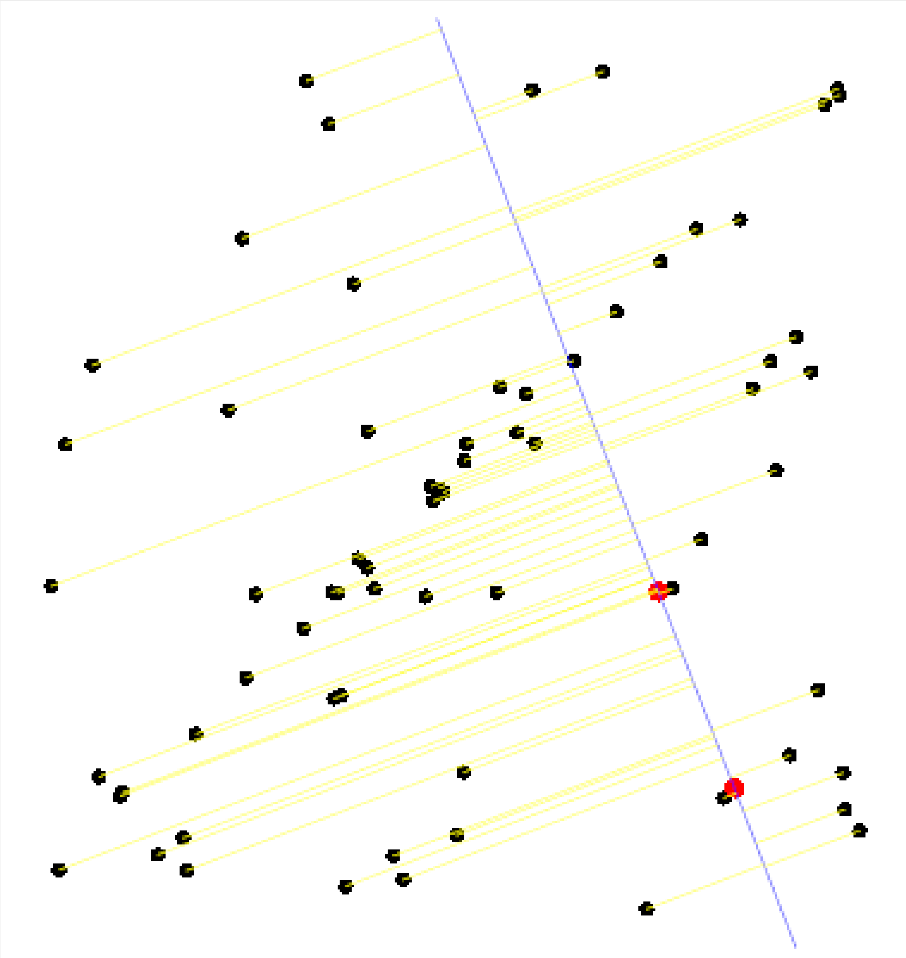
Example: Line Detection with RANSAC (3)

- Estimate a line based of the chosen points



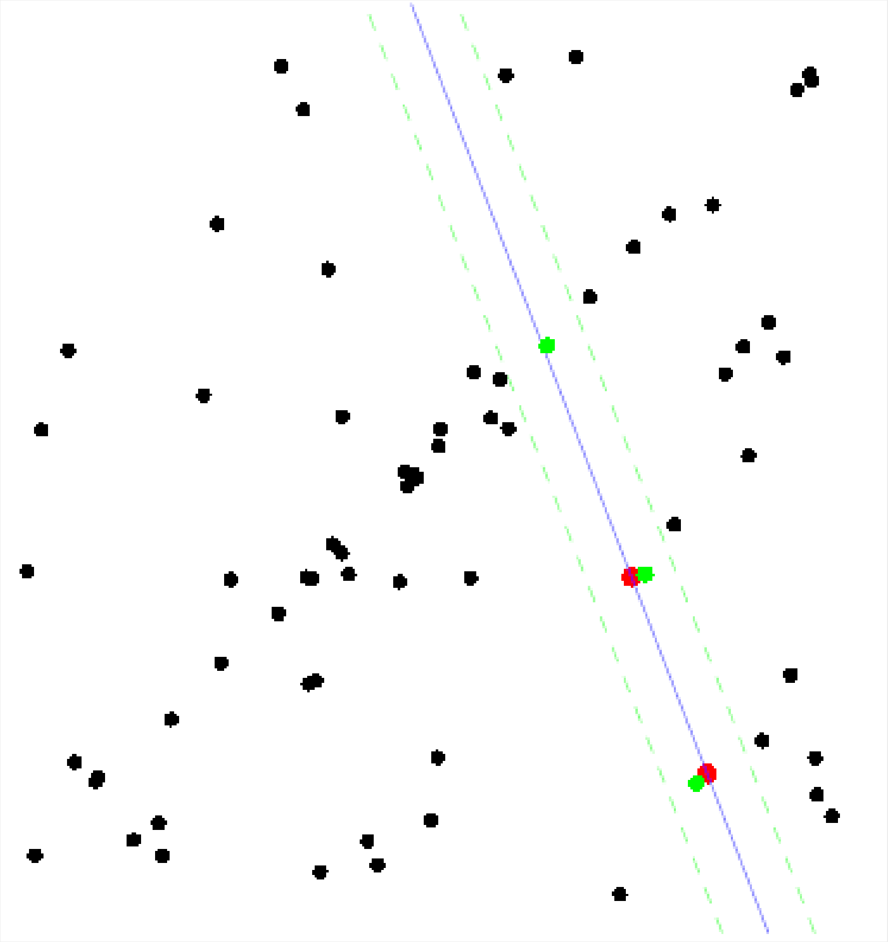
Example: Line Detection with RANSAC (4)

- Calculate the error that is made in this estimation



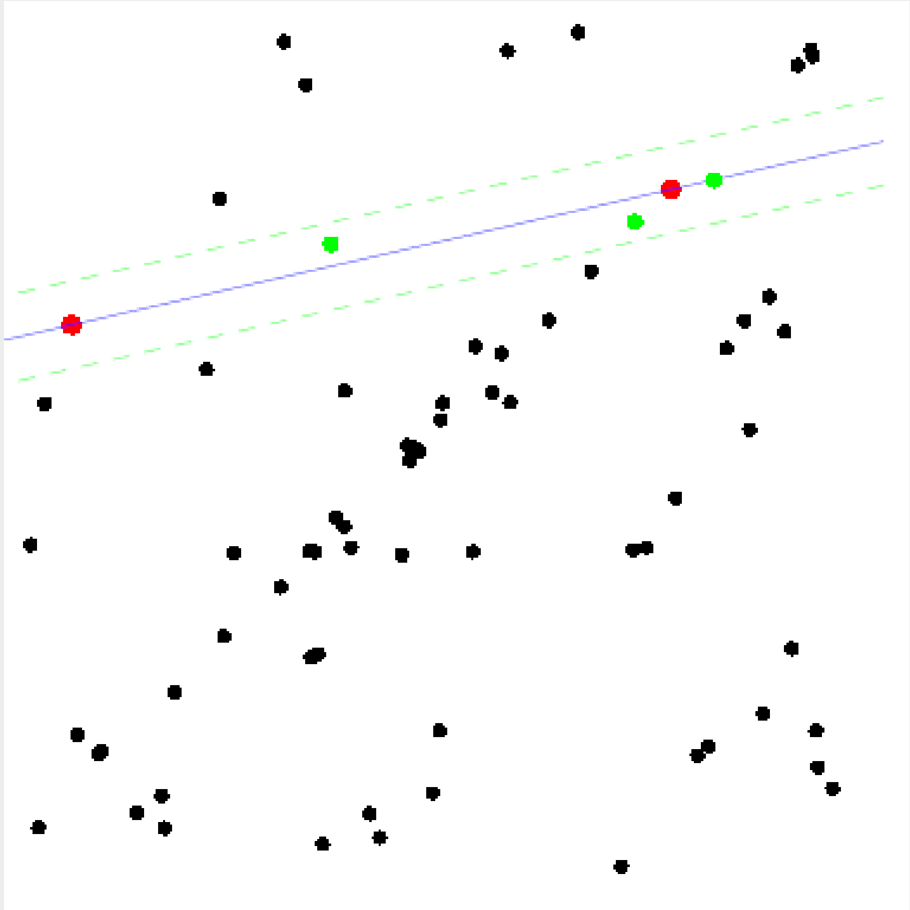
Example: Line Detection with RANSAC (5)

- Apply a threshold; estimate a new line based on the red and green points

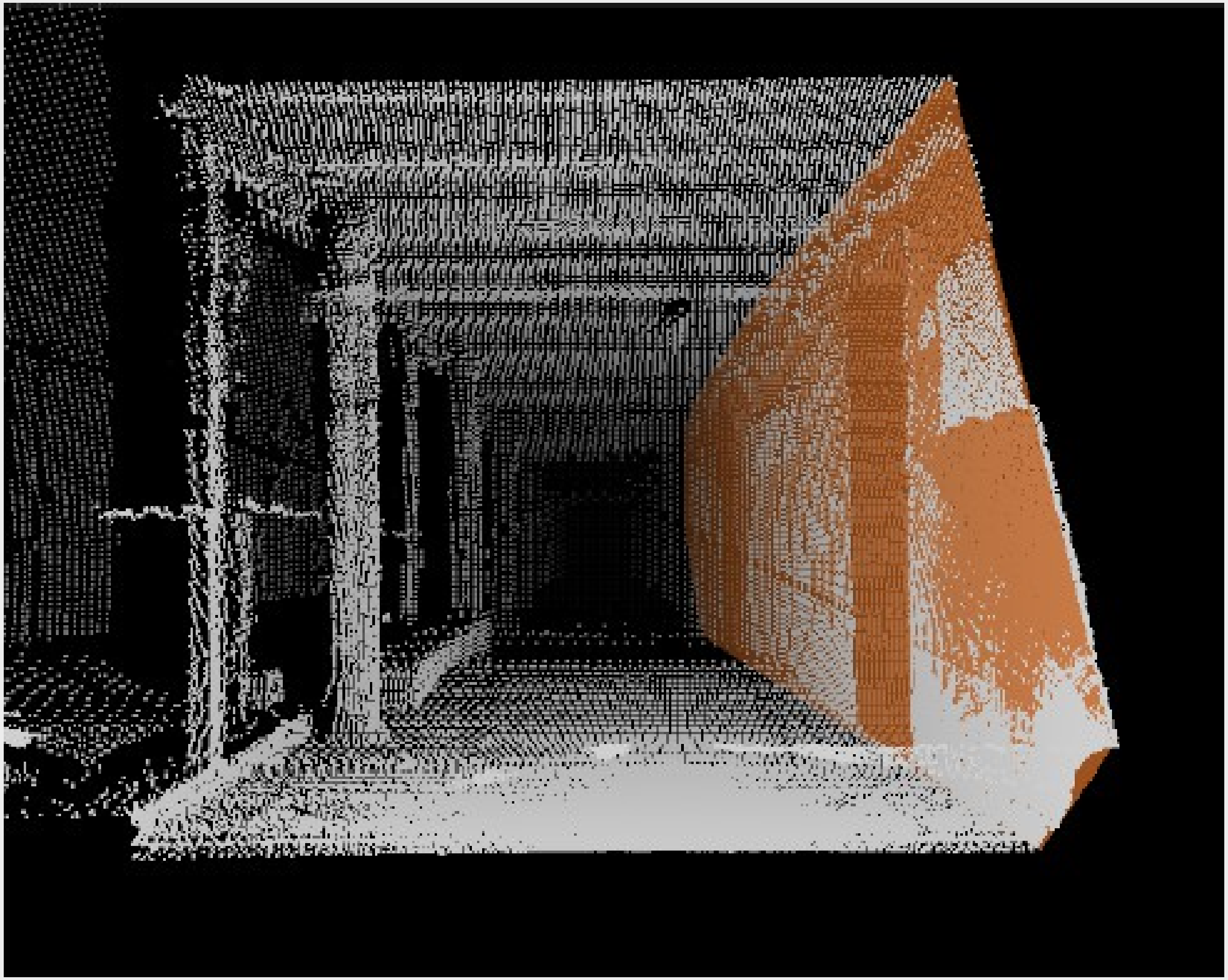


Example: Line Detection with RANSAC (7)

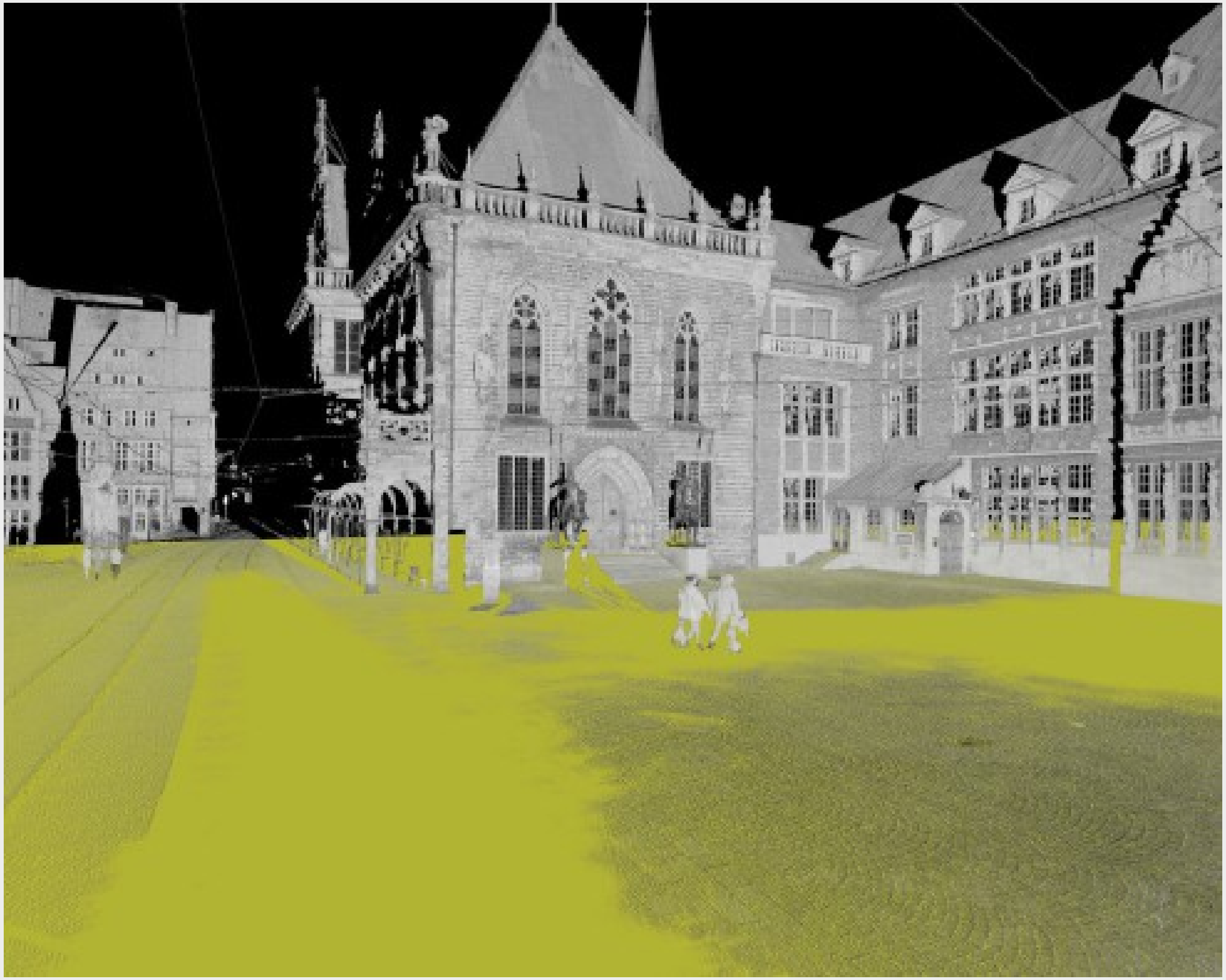
- Repeat everything for different random points



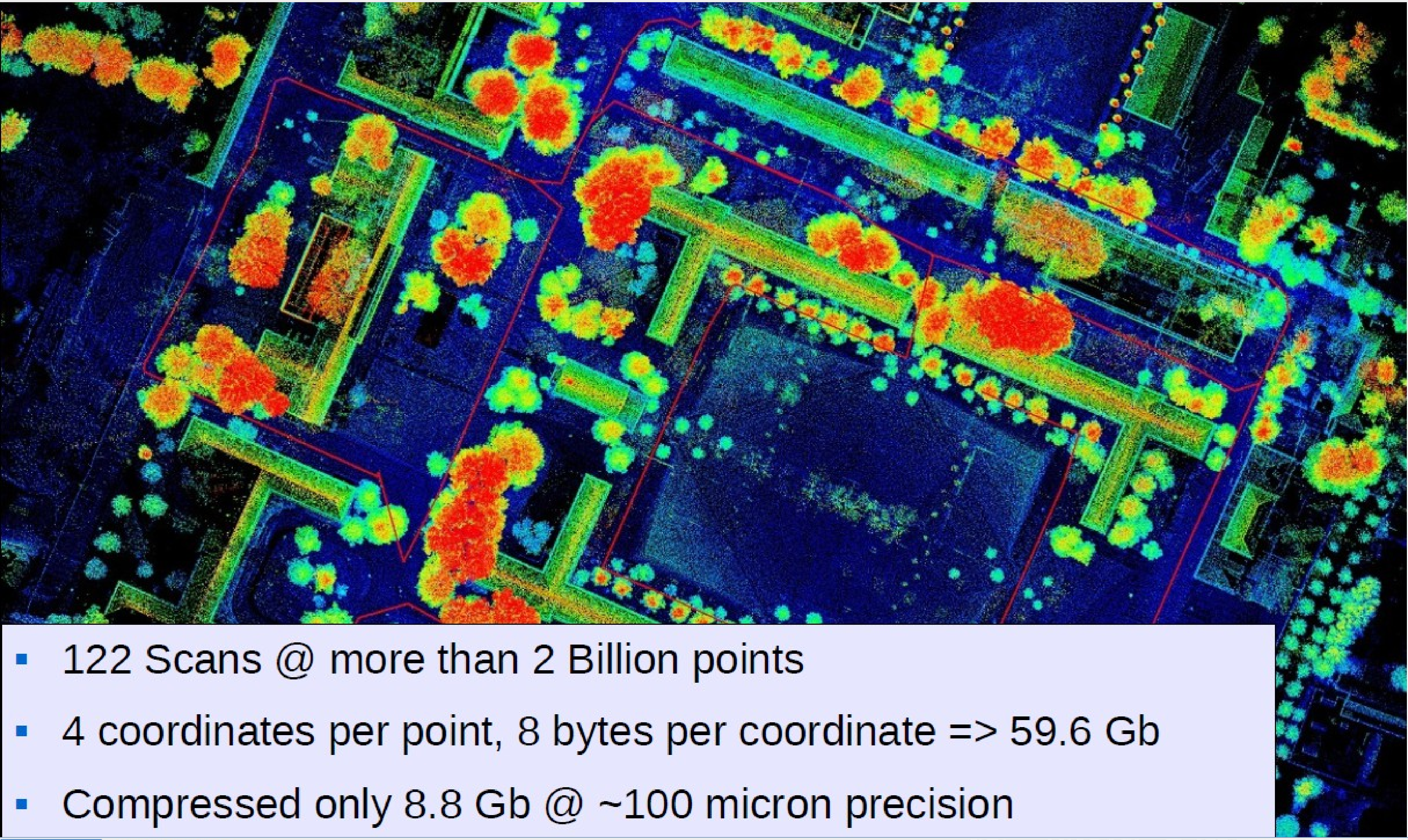
RANSAC for Plane Detection



RANSAC for Plane Detection



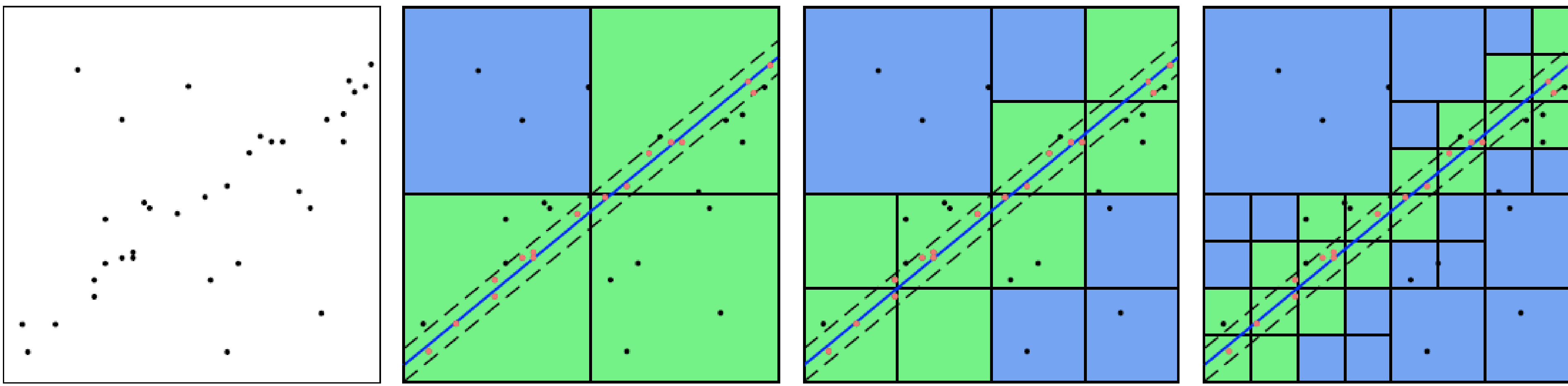
RANSAC for Plane Detection



- 122 Scans @ more than 2 Billion points
- 4 coordinates per point, 8 bytes per coordinate => 59.6 Gb
- Compressed only 8.8 Gb @ ~100 micron precision

Detecting Shapes with RANSAC

- Improve selection of sample points
  - Choose points with higher likelihood if in close proximity
  - Lower number of draws required
- Speed up validation of hypothesis



AVERAGE COMPUTING TIME IN *ms* OF RANSAC.

Data set	no octree	octree	speedup
Kurt3D	1666.57	176.69	9.43
Kinect	6905.94	429.32	16.08
city	388551.55	11084.81	35.05

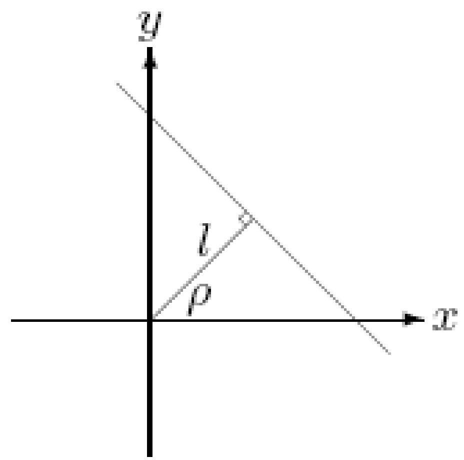
R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 2007.

Hough Transformation: Detection of Lines (1)

- Example: Accumulator with point pairs

Representation using the Hessian normal form:

$$l = x \cos \rho + y \sin \rho; \quad 0 \leq \rho < 2\pi, \; l \geq 0$$



We need a discrete accumulator *H[ρ][l]* : Both *ρ* and *l* are represented with finite

$$\begin{aligned} \rho &= 0, \; \Delta\rho, \; 2\Delta\rho, \; \ldots \\ l &= 0, \; \Delta l, \; 2\Delta l, \; \ldots \end{aligned}$$

Hough Transformation: Detection of Lines (2)

Hough Transformation of a line
(based on edge point pairs)

- Input: edge points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Output: lines that go through these edge points

```
Set all elements of the accumulator H[ρ][l] to zero;
for (all pairs of edge points in k(r, c)) {
      calculate the corresponding ρ and l;
      H[ρ][l] ++;  /* Consider discretization of ρ and l */
}
Detect peaks in H[ρ][l];
```

- One can expect that there will be peaks in the accumulator array H[ρ][l]. All peaks correspond to a line in the image.
- Huge number of edge point pairs: O(n^2)

Hough Transformation: Detection of Lines (3)

- Accumulator with single edge points:

  Reduction of the work load: Construct the Accumulator H[ρ][l] with "hints" to possible lines based on single edge points. Unfortunately, every edge point (x_i, y_i) implies not one edge, but a set of edges (ρ, l):

  $$l = x_i \cos \rho + y_i \sin \rho$$

  i.e., a sinusoidal curve in the ρl space.

- Solution: All possible parameters (ρ, l), that fulfill the above constraint (I.e. they represent a line with the current edge point (x_i, y_i)) are considered and the corresponding counter in the array is incremented.

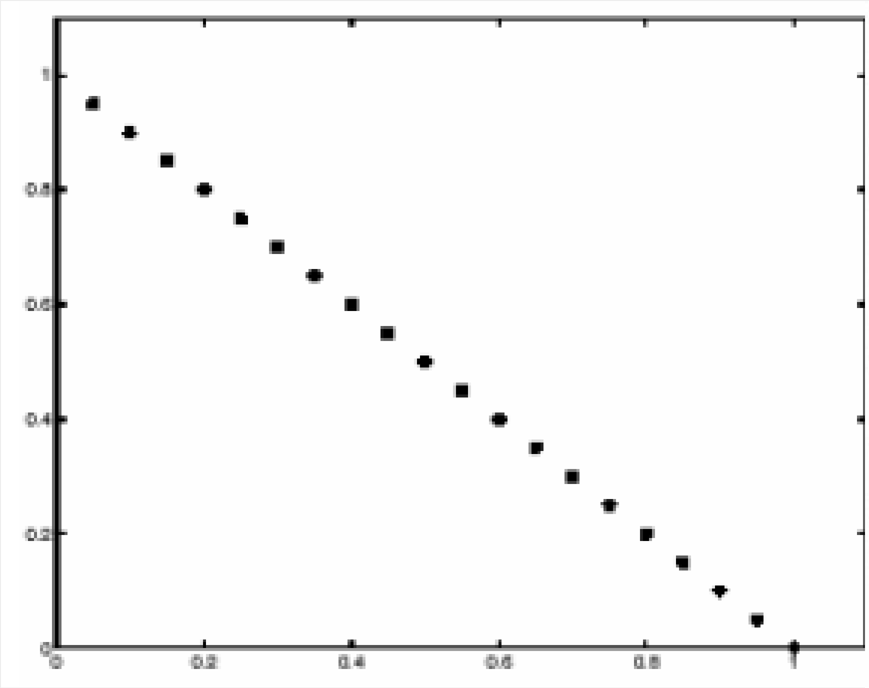
Hough Transformation: Detection of Lines (4)

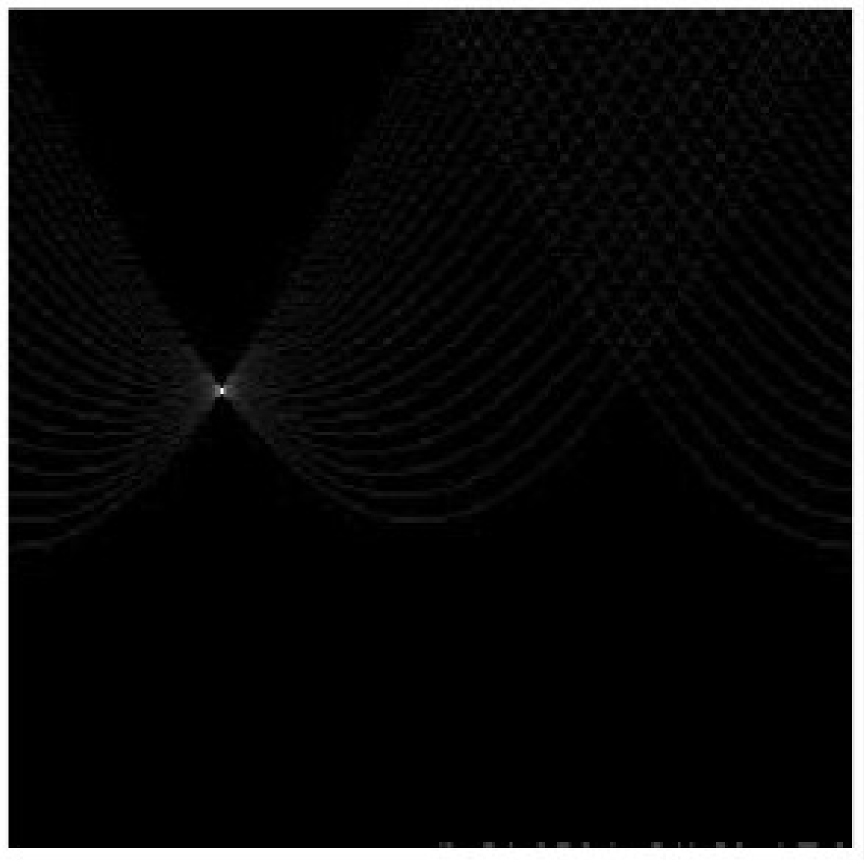
- Input: Edge points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Output: Lines, that go though these edge points

```
1. Transform all adge points (xi, yi) according to
   l = xi cos ρ + yi sin ρ from the xy-space to the ρl-space.
   for (ρ = 0; ρ < 2π; ρ+=dρ)
     for (l = 0; l < lmax; l+=dl)
        H[ρ][l] = 0;
   for (i = 1; i ≤ n; i++)
     for (ρ = 0; ρ < 2π; ρ+=dρ) {
        l = xi cos ρ + yi sin ρ;
        H[ρ][l] ++;  /* Regard the discretisation of l */
                     /* Increment according to edge width s(xi, yi) poss. */
     }
2. Search for cluster points in ρl-space, i.e. in H[ρ][l].
3. All cluster points (ρ0, l0) define a line
   l0 = x cos ρ0+y sin ρ0 in xy-space.
```

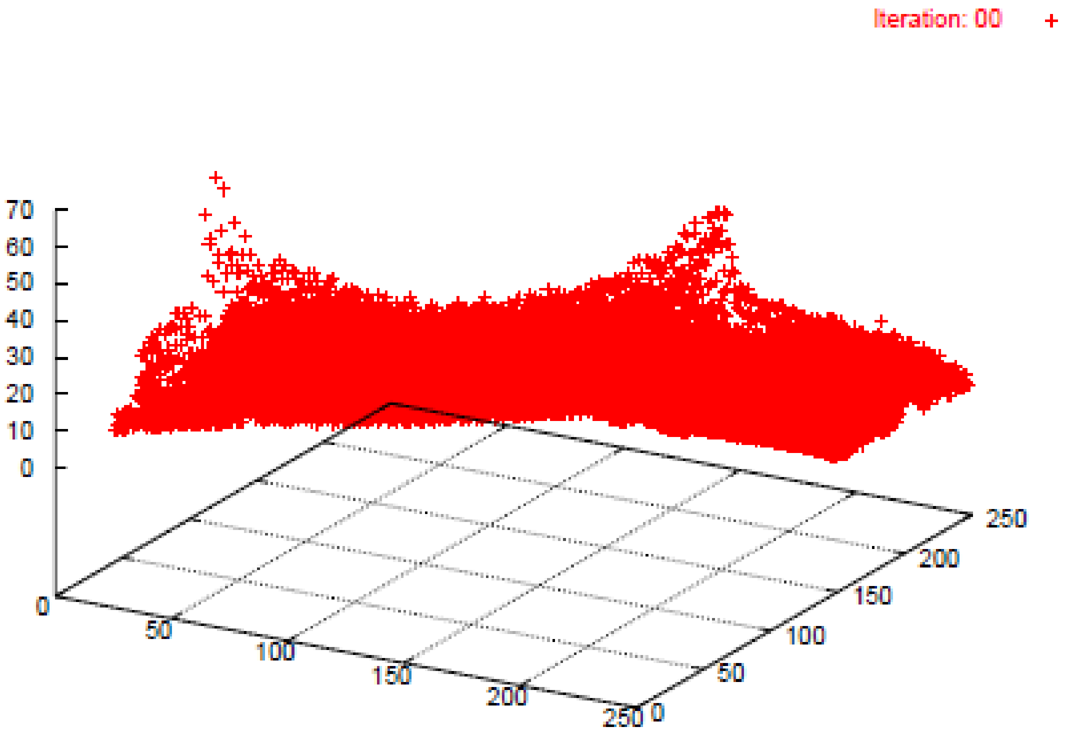
Hough Transformation: Detection of Lines (5)

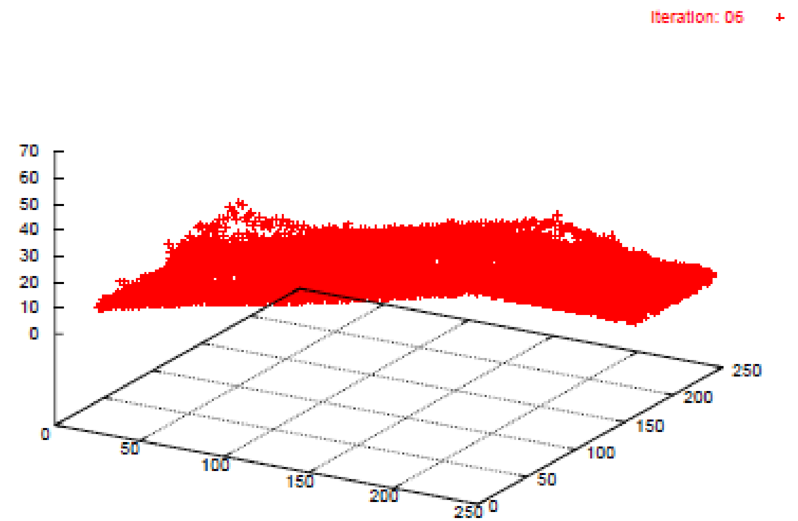
- Example:

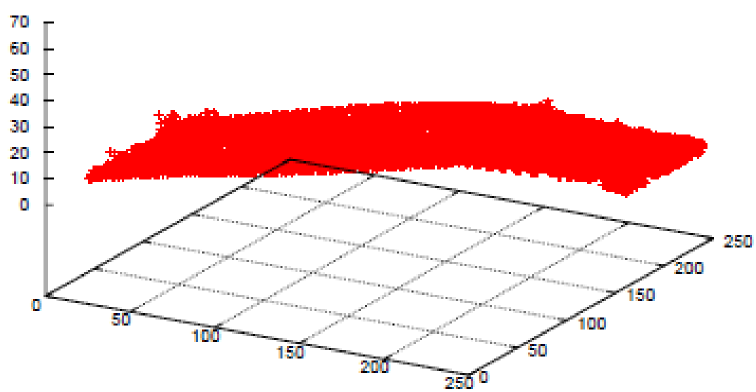


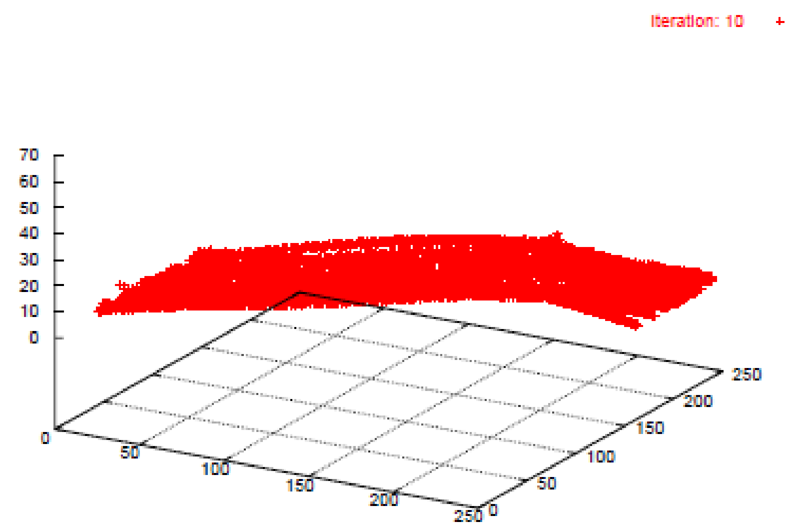


Hough Transformation: Iterations (1)

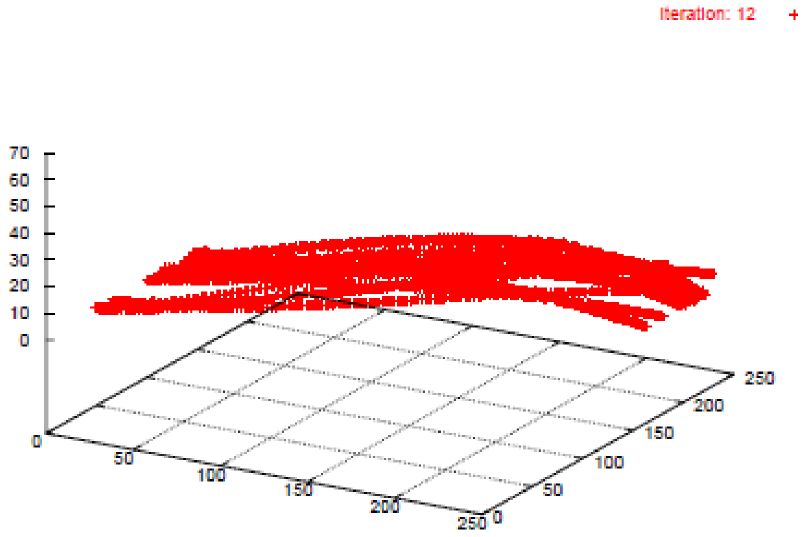


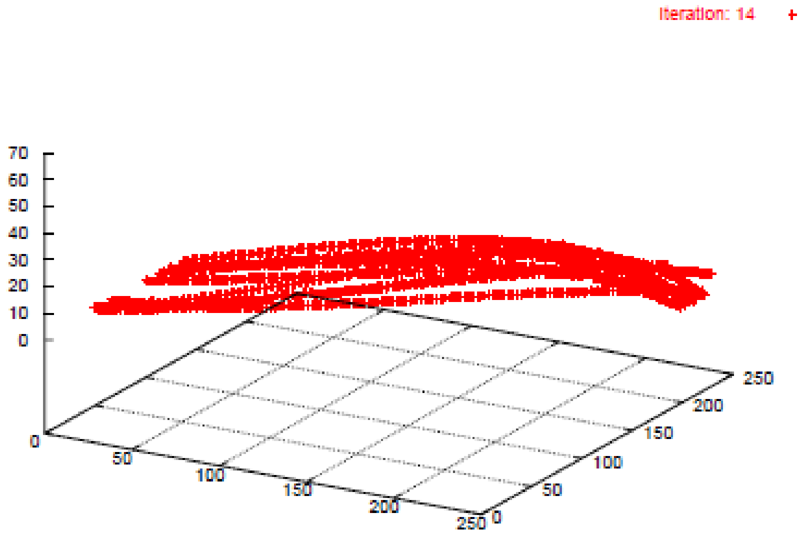


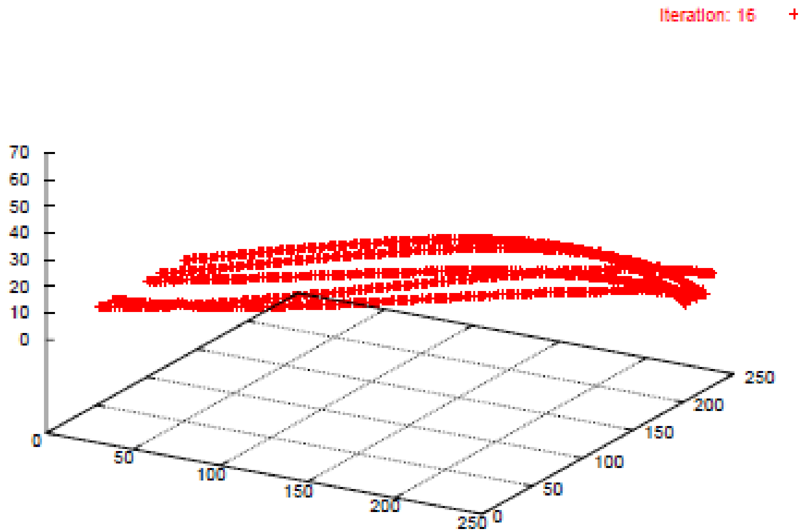


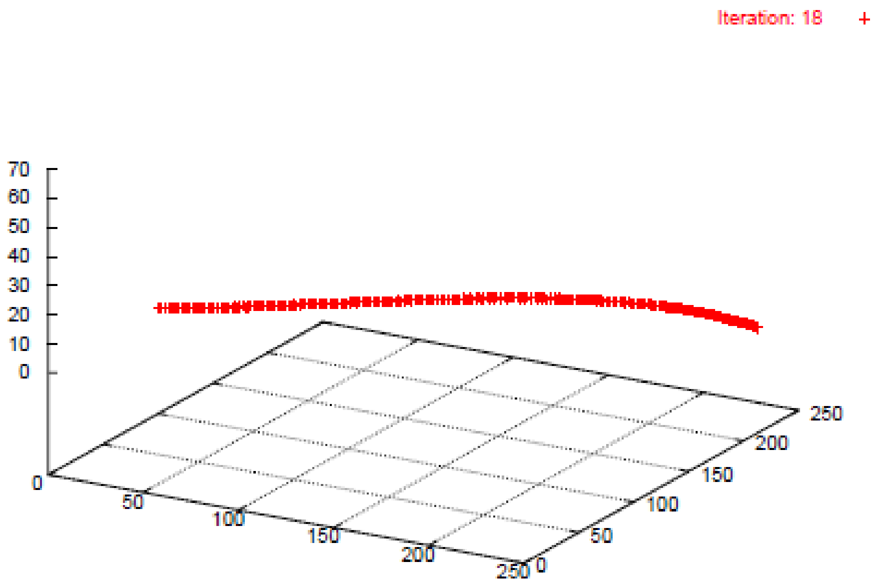


Hough Transformation: Iterations (2)

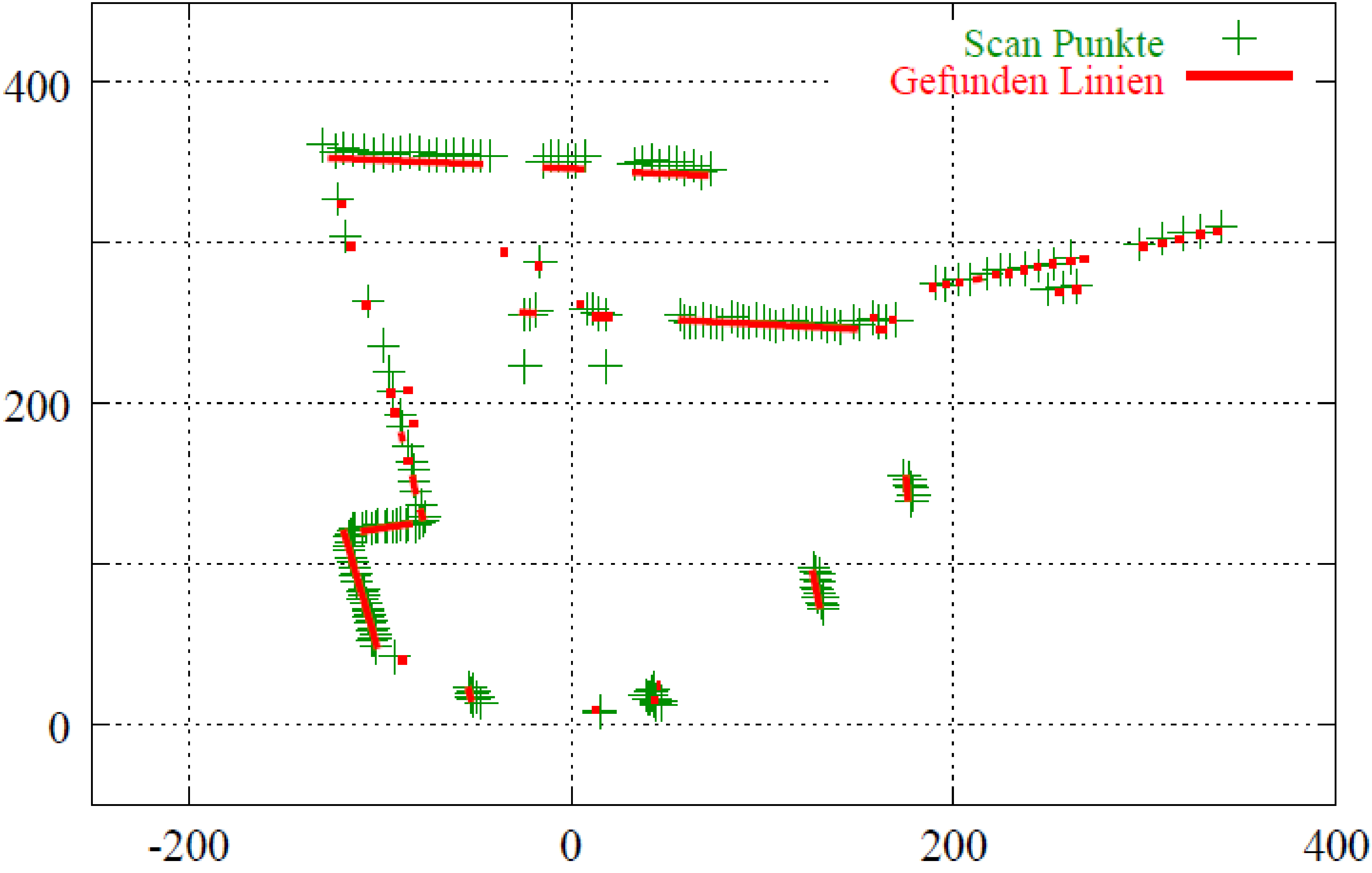






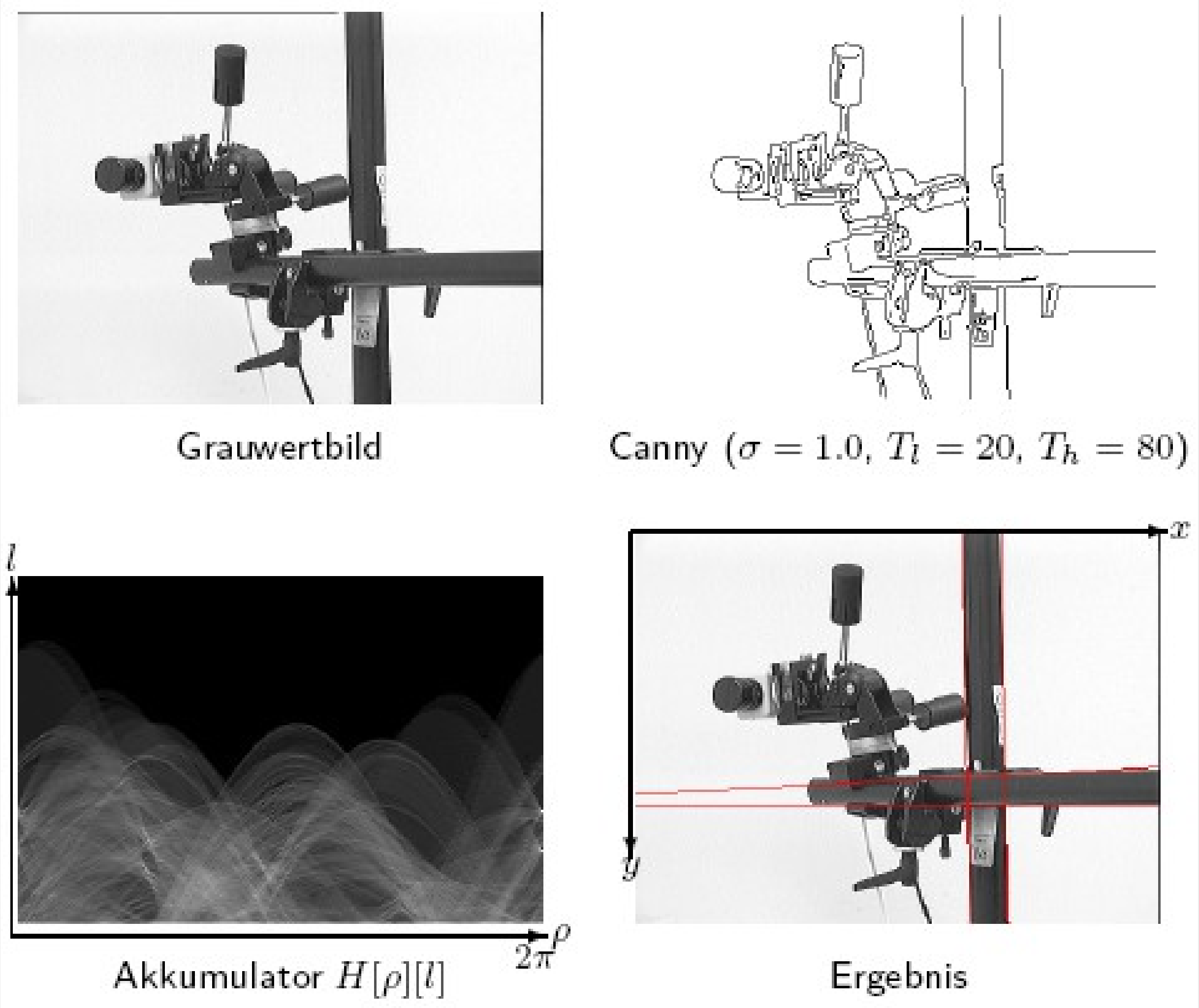


Hough Transformation: Results



Hough Transformation: Example

- Detection of Lines:

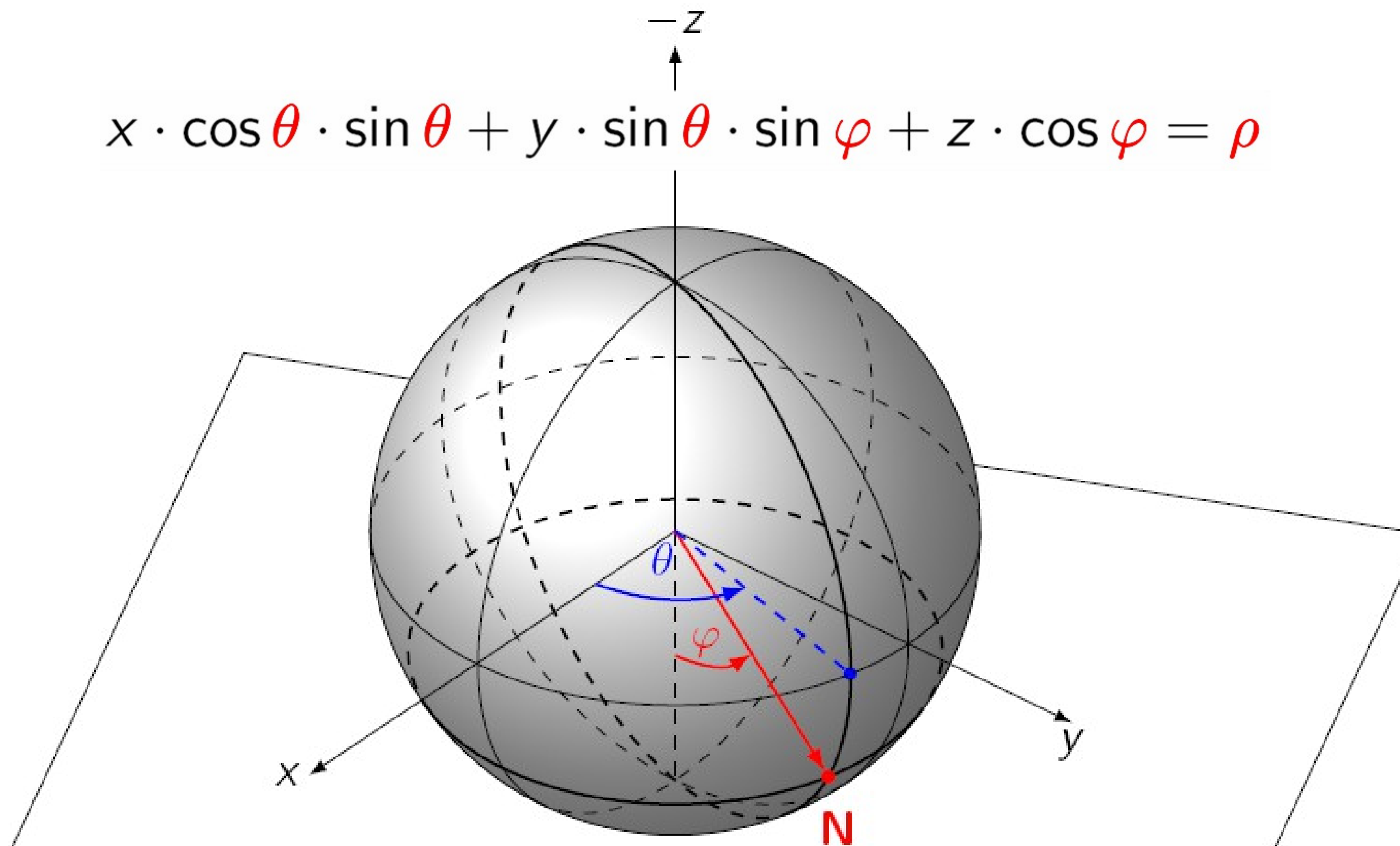


From 2D to 3D – From Lines to Planes

$$\rho = \mathbf{p} \cdot \mathbf{n} = p_x n_x + p_y n_y + p_z n_z = \rho$$

Polar Coordinates

$$x \cdot \cos\theta \cdot \sin\theta + y \cdot \sin\theta \cdot \sin\varphi + z \cdot \cos\varphi = \rho$$

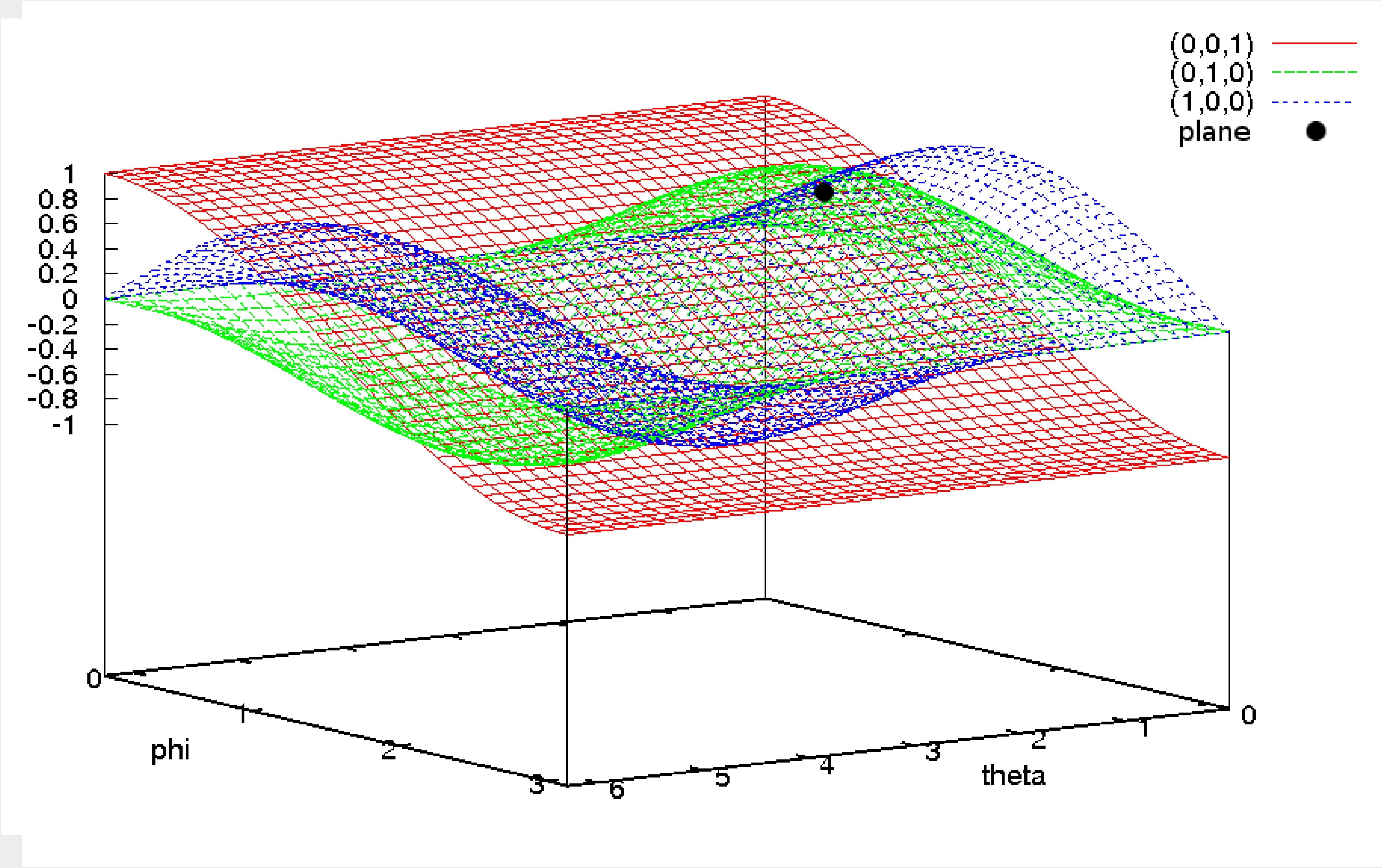


Hough Transform (HT)

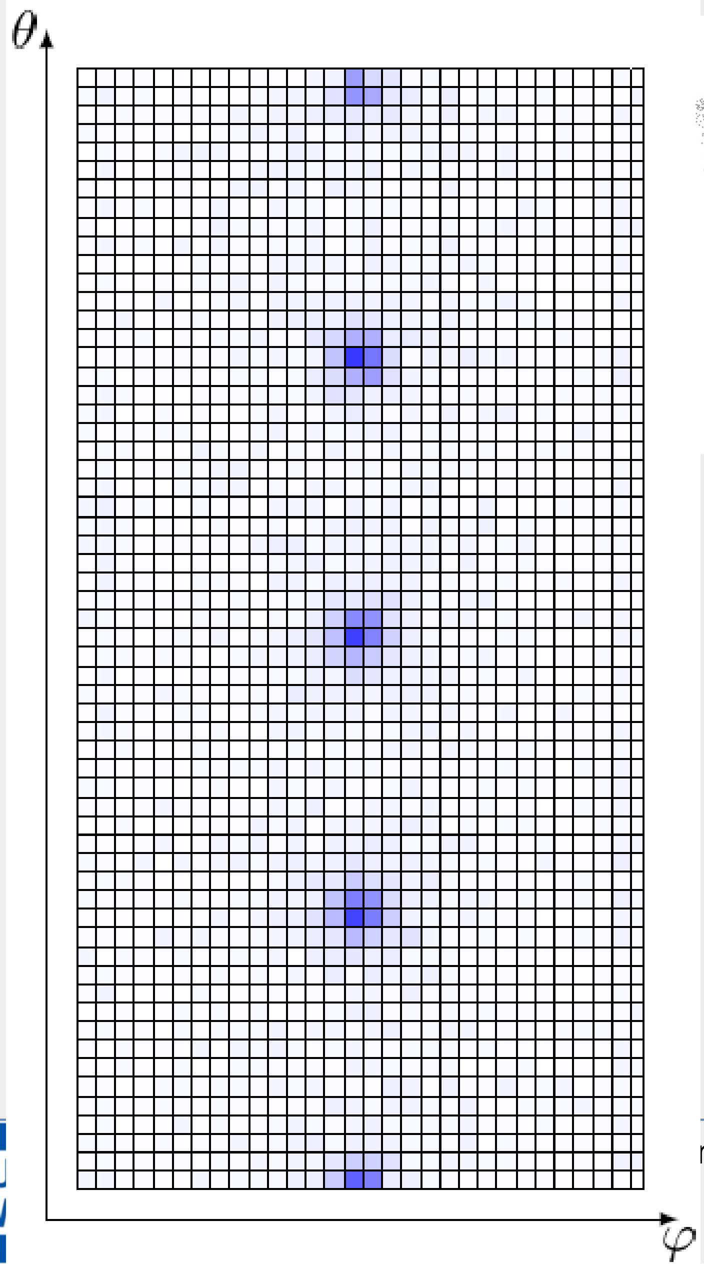
$$x \cdot \cos\theta \cdot \sin\theta + y \cdot \sin\theta \cdot \sin\varphi + z \cdot \cos\varphi = \rho$$

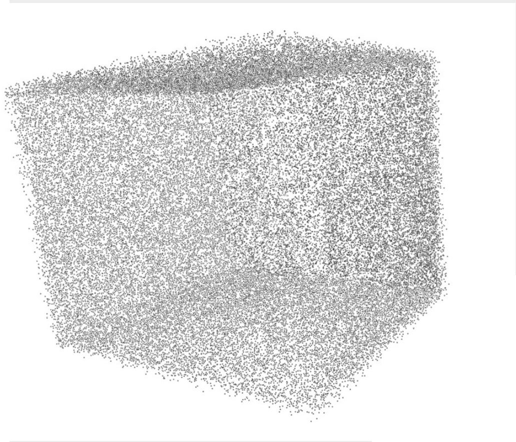
- Hough Space: (φ, θ, ρ) Space
  - $0 < \varphi < \pi$
  - $0 < \theta < 2\pi$
  - ρ ??
- Cartesian Space: (x,y,z) – Space
  - -∞ < x,y,z < ∞
- Hough Transform
  - Cartesian Space → Hough Space
  - For a point (x,y,z) HT yields all planes (φ, θ, ρ) that go through (x,y,z)

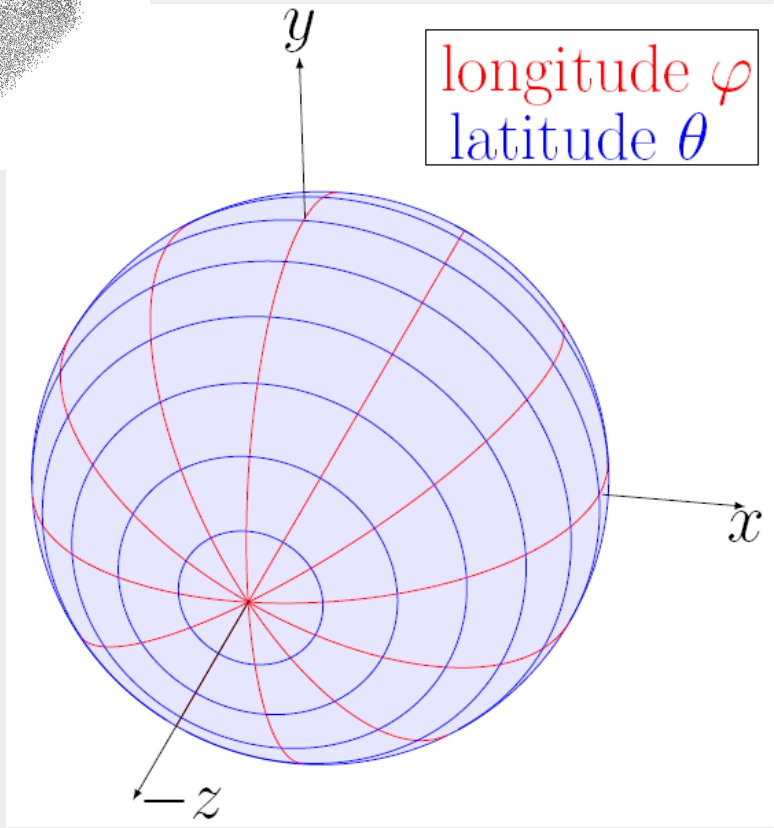
Hough Transform – Example



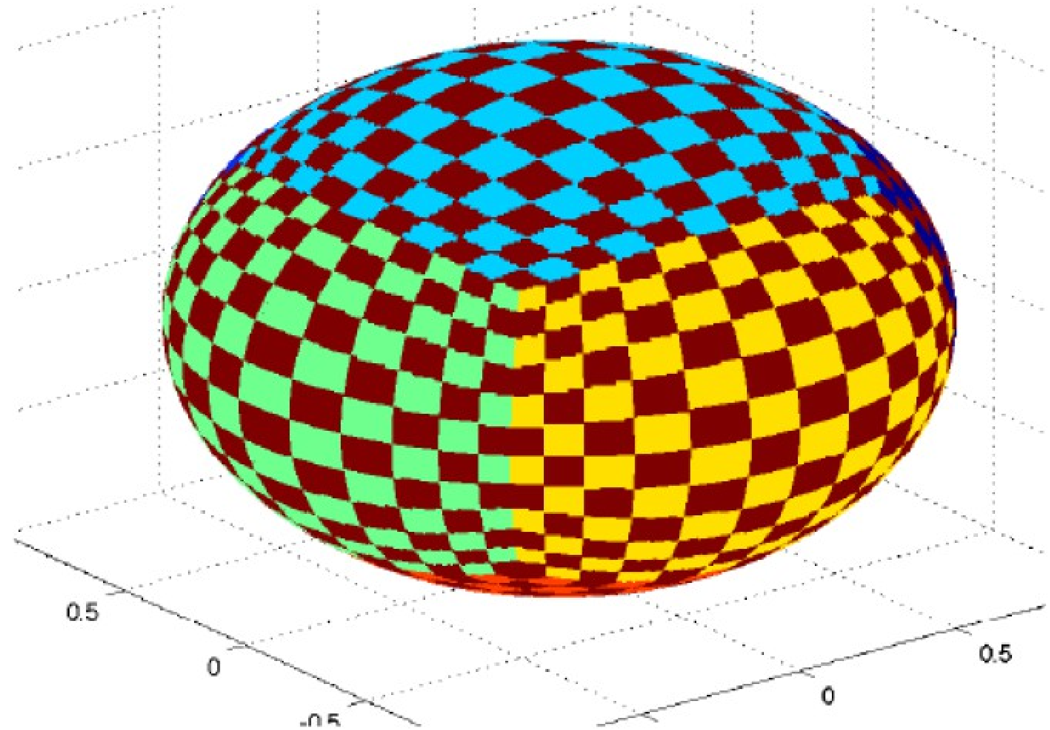
Discretization – Accumulator Array

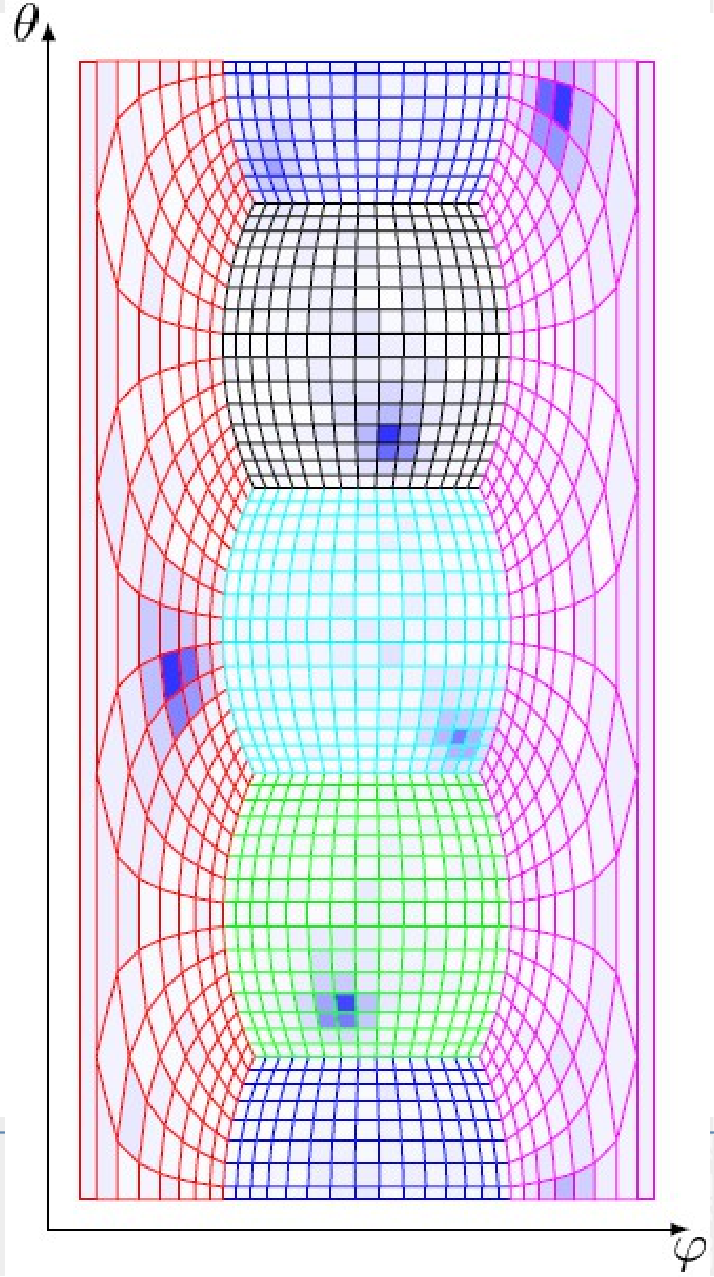






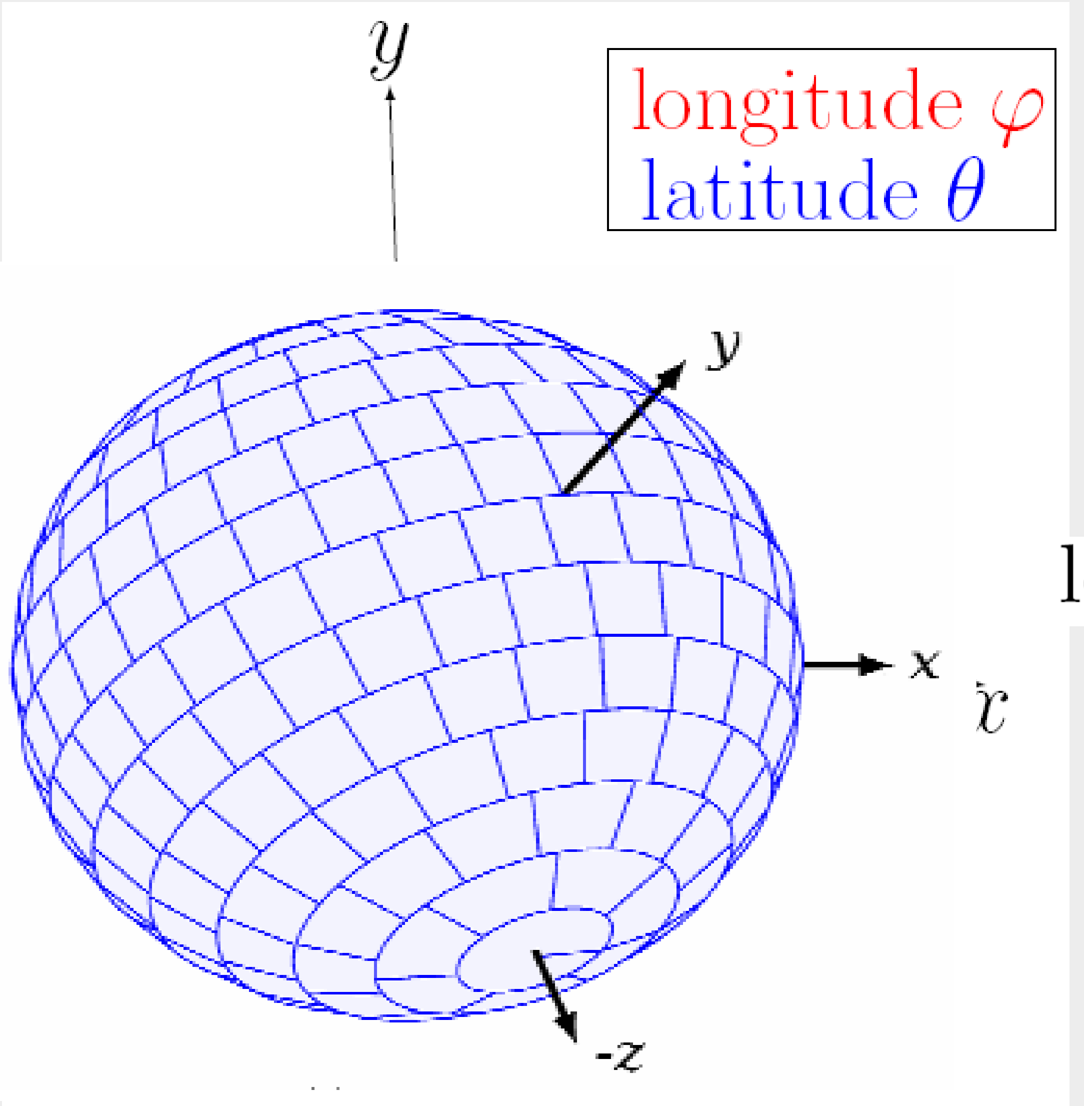
Discretization – Related Work





[Censi, 2004]

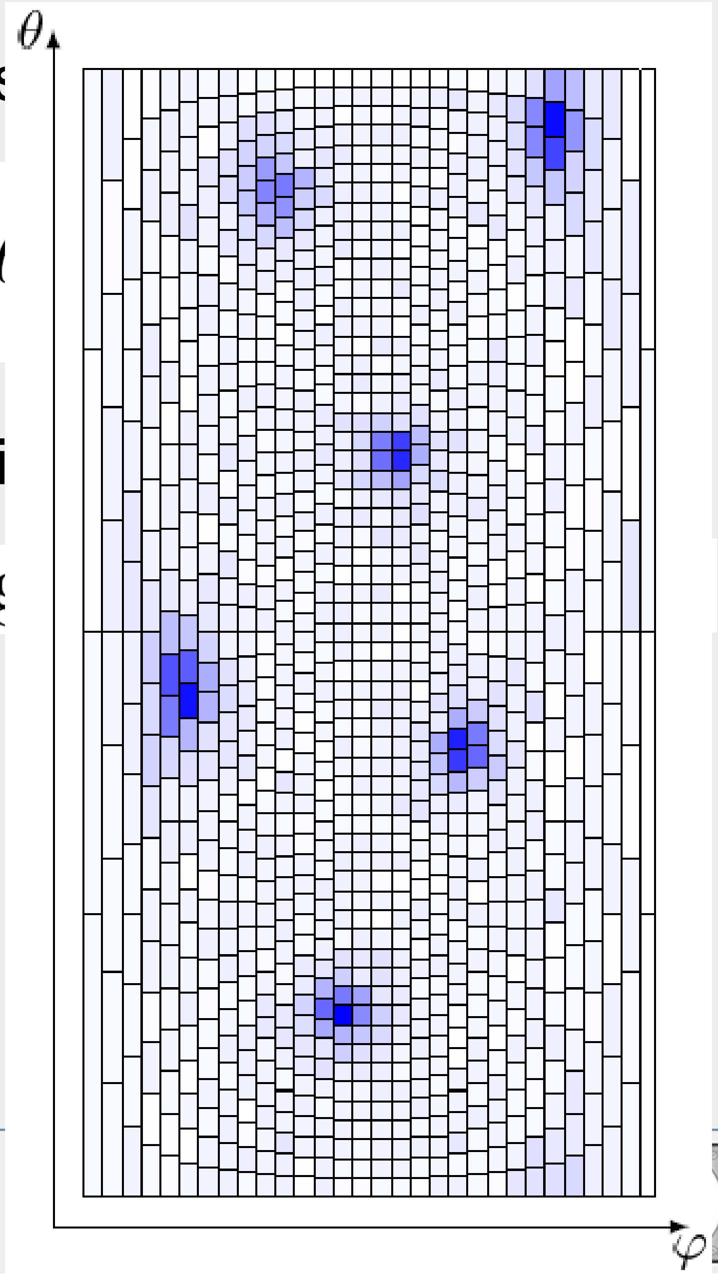
Accumulator Ball



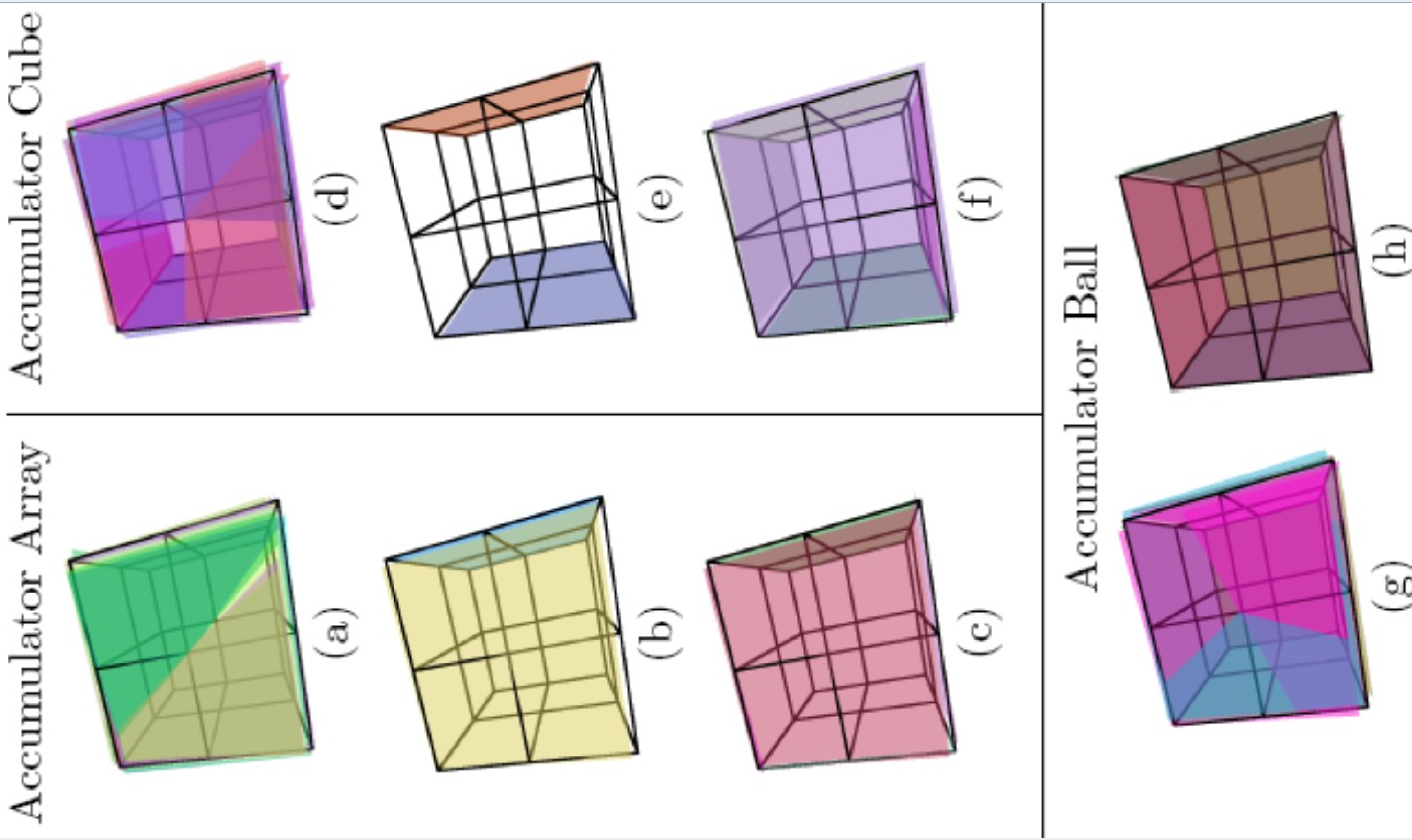
• dis

wi

leng



Comparing Accumulators (1)



Hough Variants - SHT

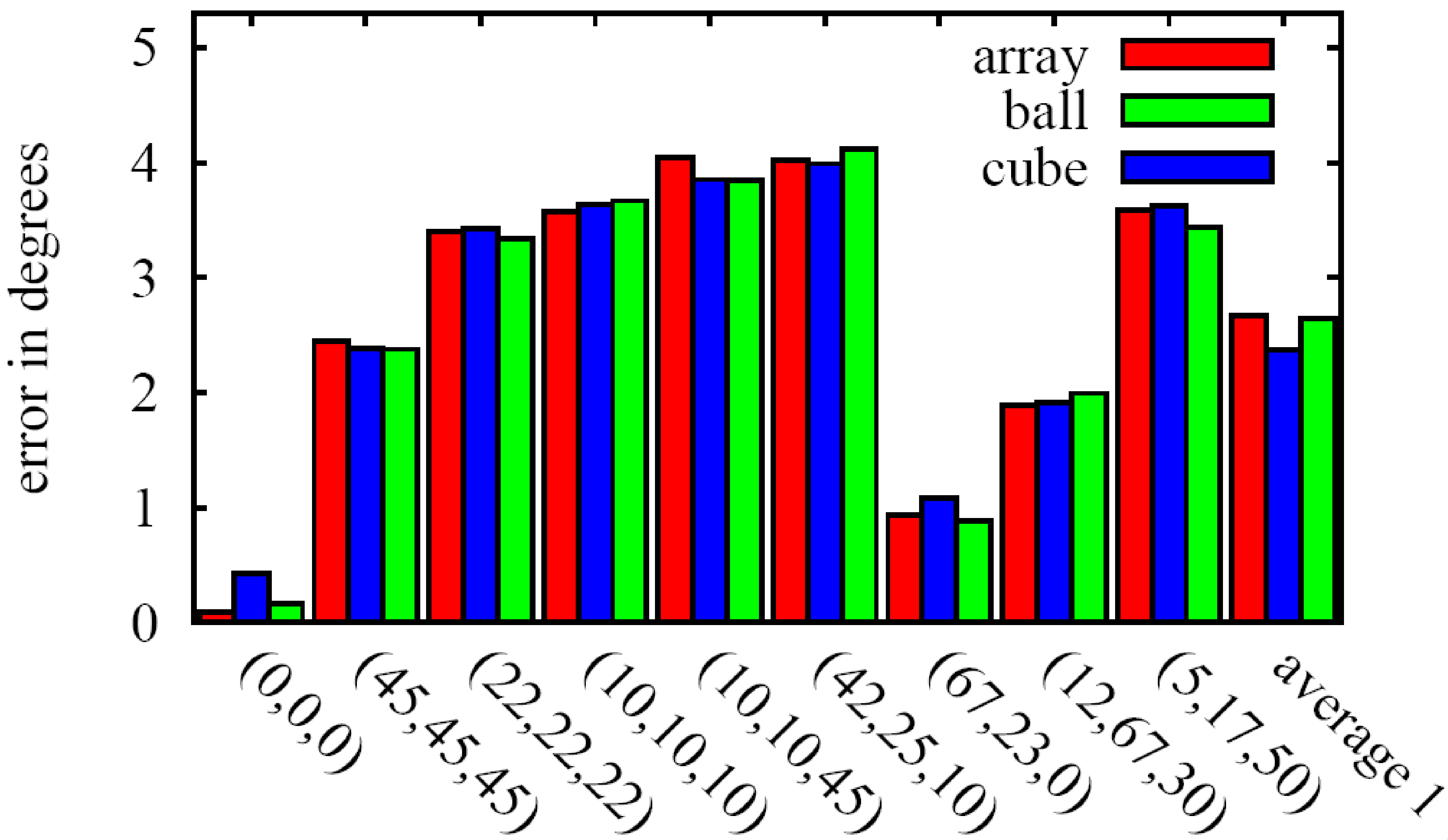
Algorithm 1 Standard Hough Transform (SHT)

1: **for all** points $\mathbf{p}_i$ in point set P **do**
2:     **for all** cells (ρ, φ, θ) in accumulator A **do**
3:         **if** point $\mathbf{p}_i$ lies on the plane defined by (ρ, φ, θ) **then**
4:             accumulate cell $A(\rho, \varphi, \theta)$
5:         **end if**
6:     **end for**
7: **end for**
8: Search for the most prominent cells in the accumulator, that define the detected planes in P

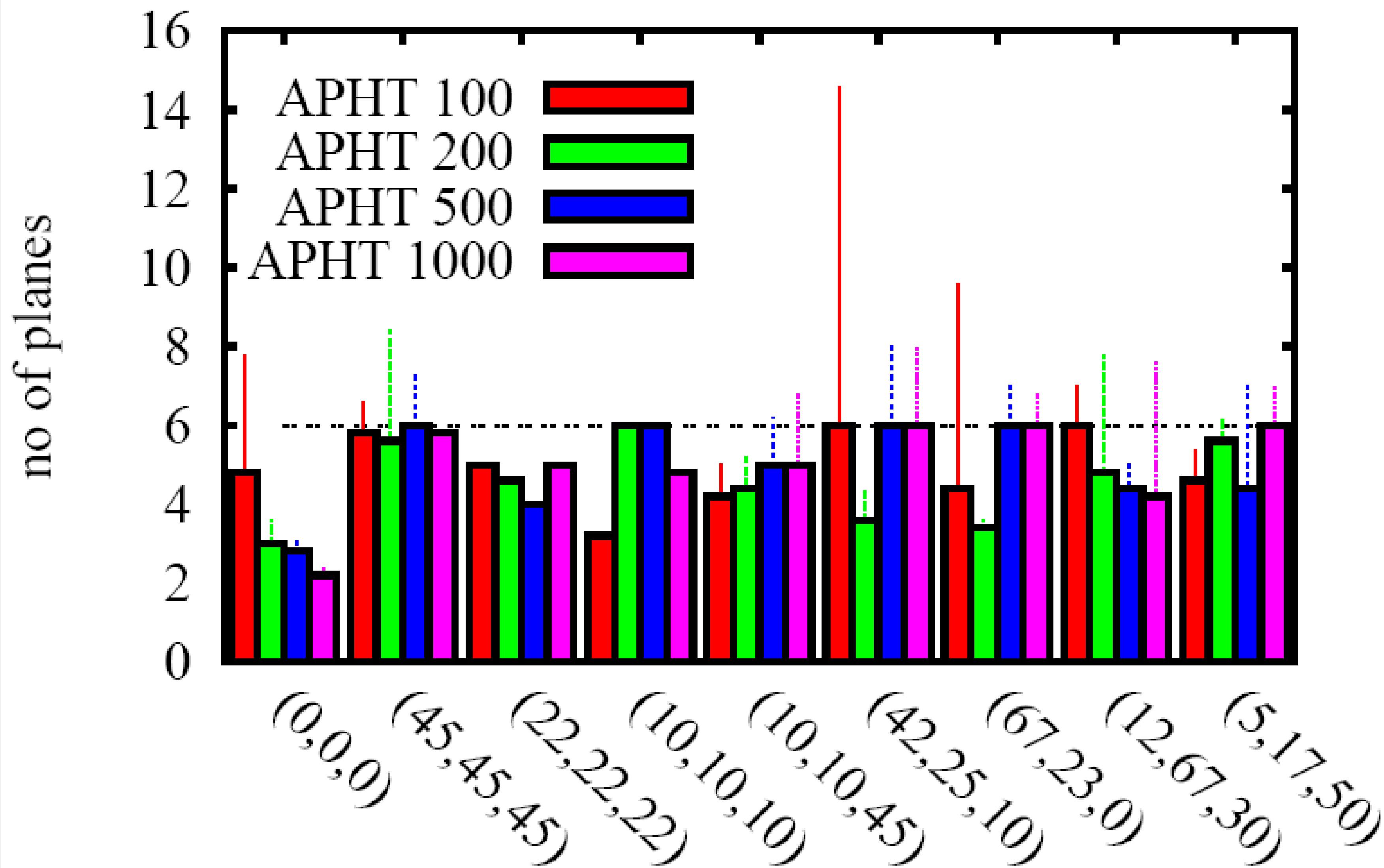
Hough Variants

- Probablistic Hough Transform (PHT)
  - Use p% of the input points only
- Progressive Probabilistic Hough Transform (PPHT)
  - Pick points randomly and perform HT
  - Quit when one cell has been voted by p% of the points
- Adaptive Probabilistic Hough Transform (APHT)
  - Pick points randomly and perform HT
  - Obtain a list of maxima
  - Quit when list of maxima remains stable
- Randomized Hough Transform (RHT)
  - Pick three points randomly
  - Accumulate the cell corresponding to the plane spanned by these points
  - Delete points of plane when threshold is reached

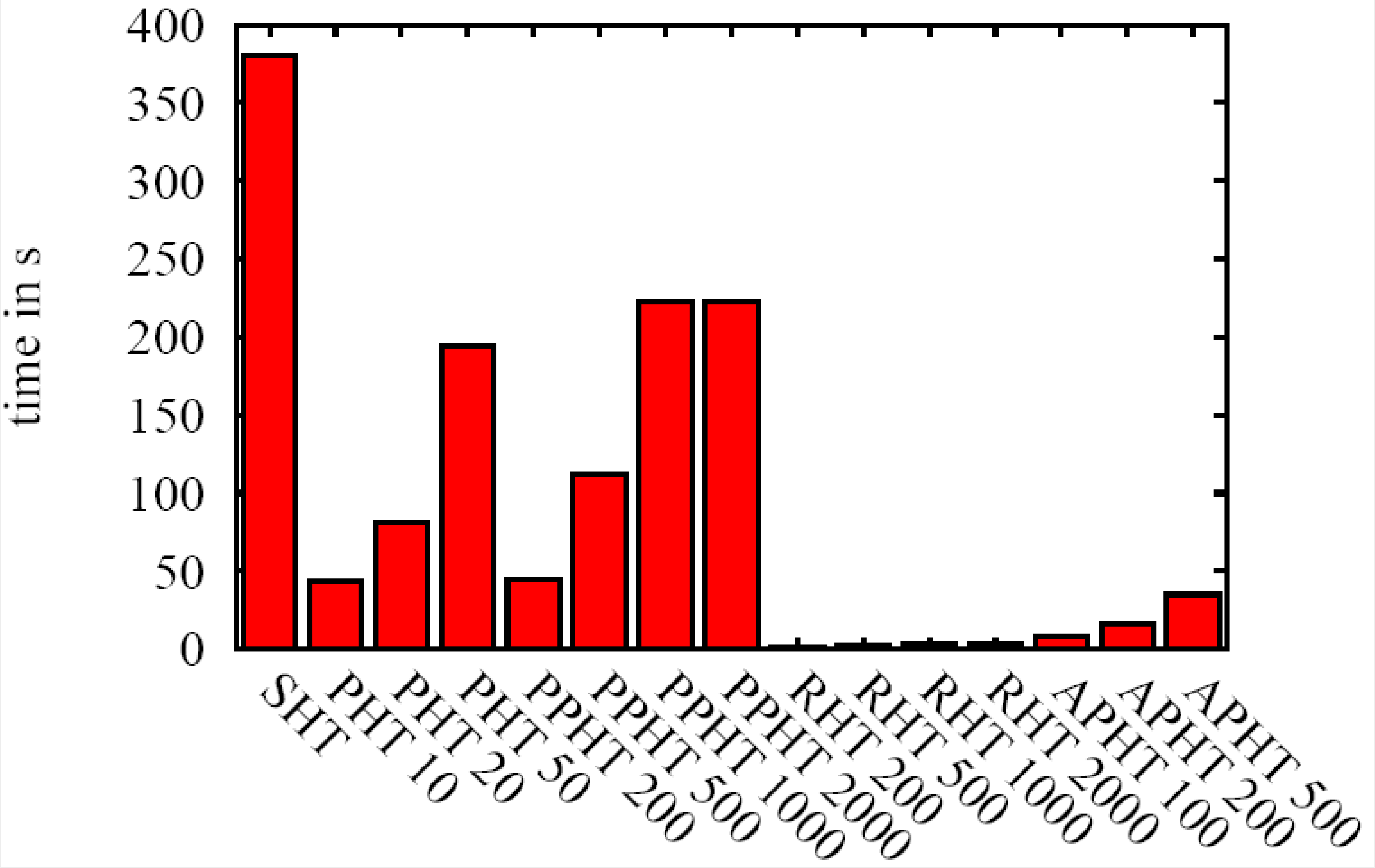
Comparing Accumulators (2)



Comparing Hough Variants (1)



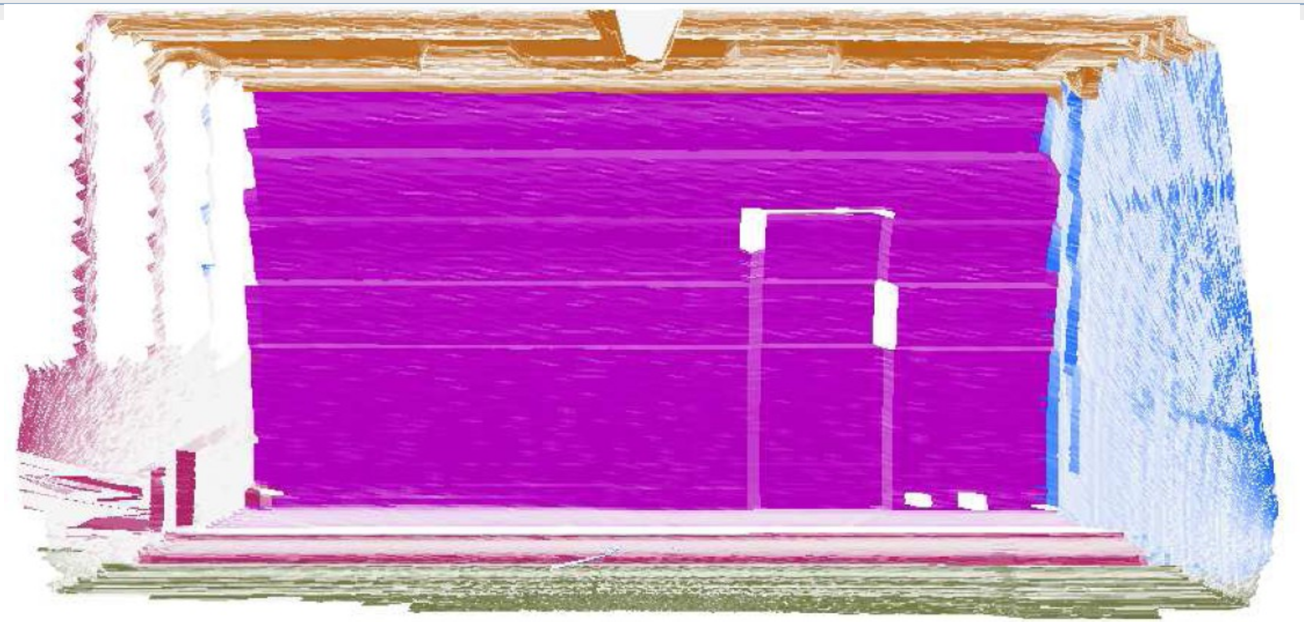
Comparing Hough Variants (2)



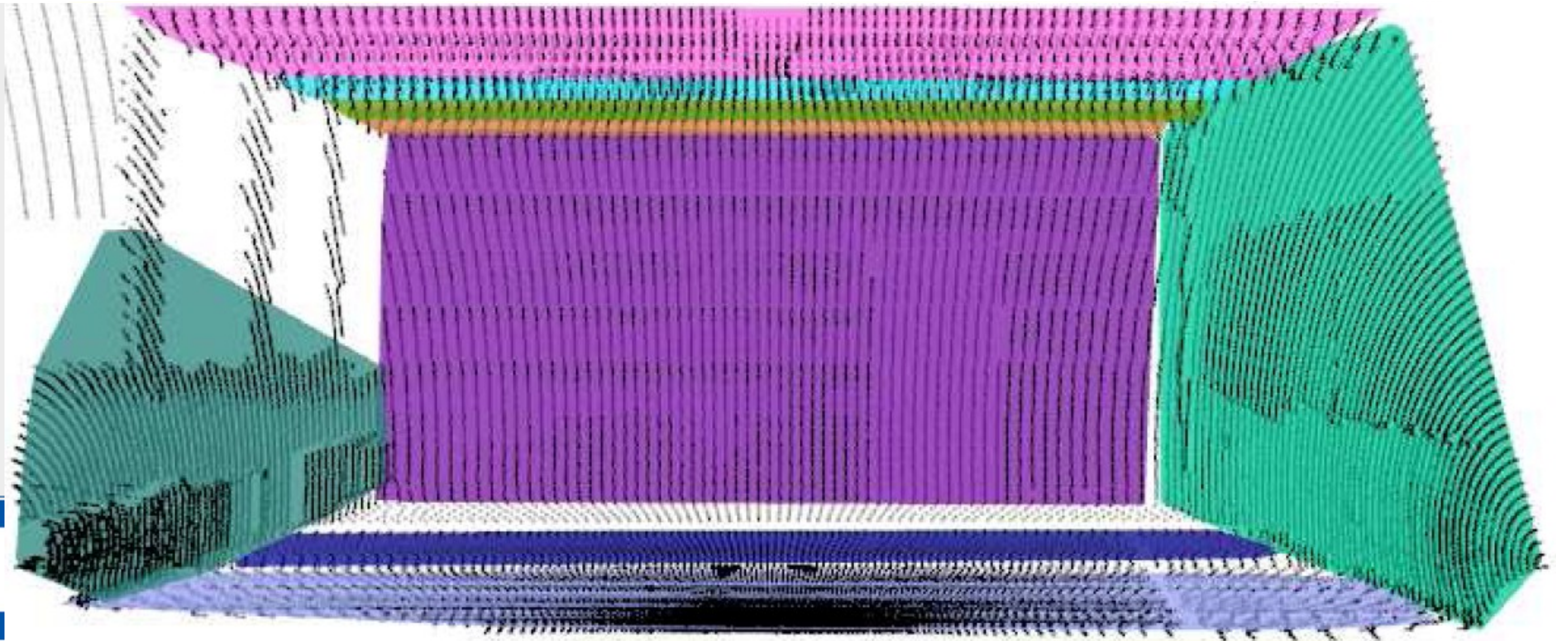
Comparison with Related Work (1)

- Randomized Hough Transform (RHT)
- Region Growing (RG) [Poppinga, 2008]
- Hierarchical Fitting Primitives (HFP) [Attene, 2006]

Comparision with Related Work (2)

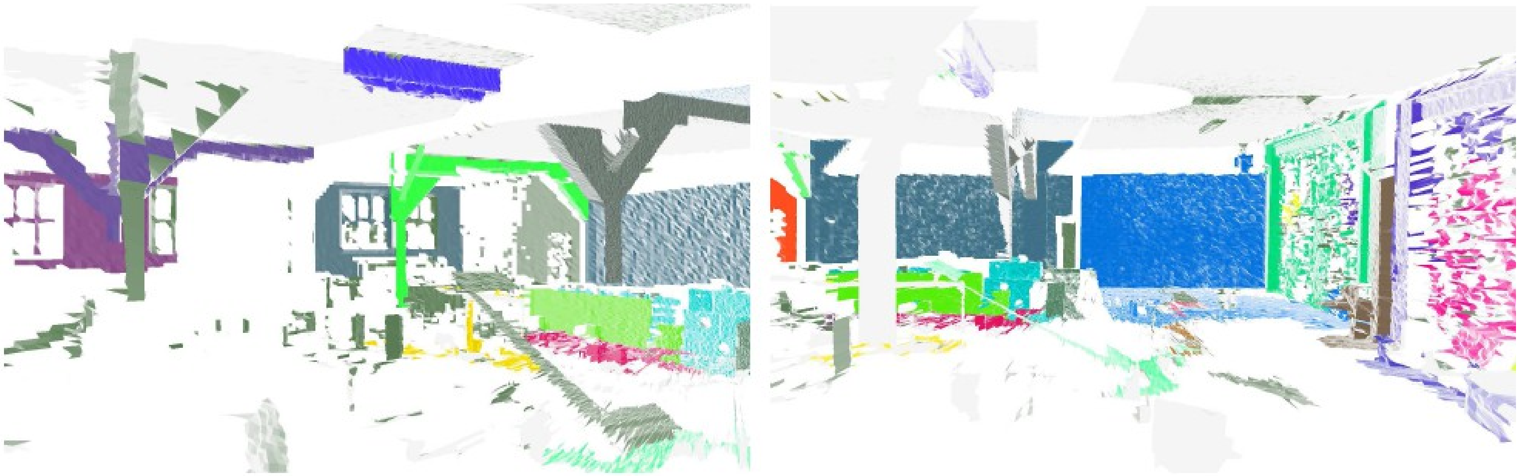


HFP

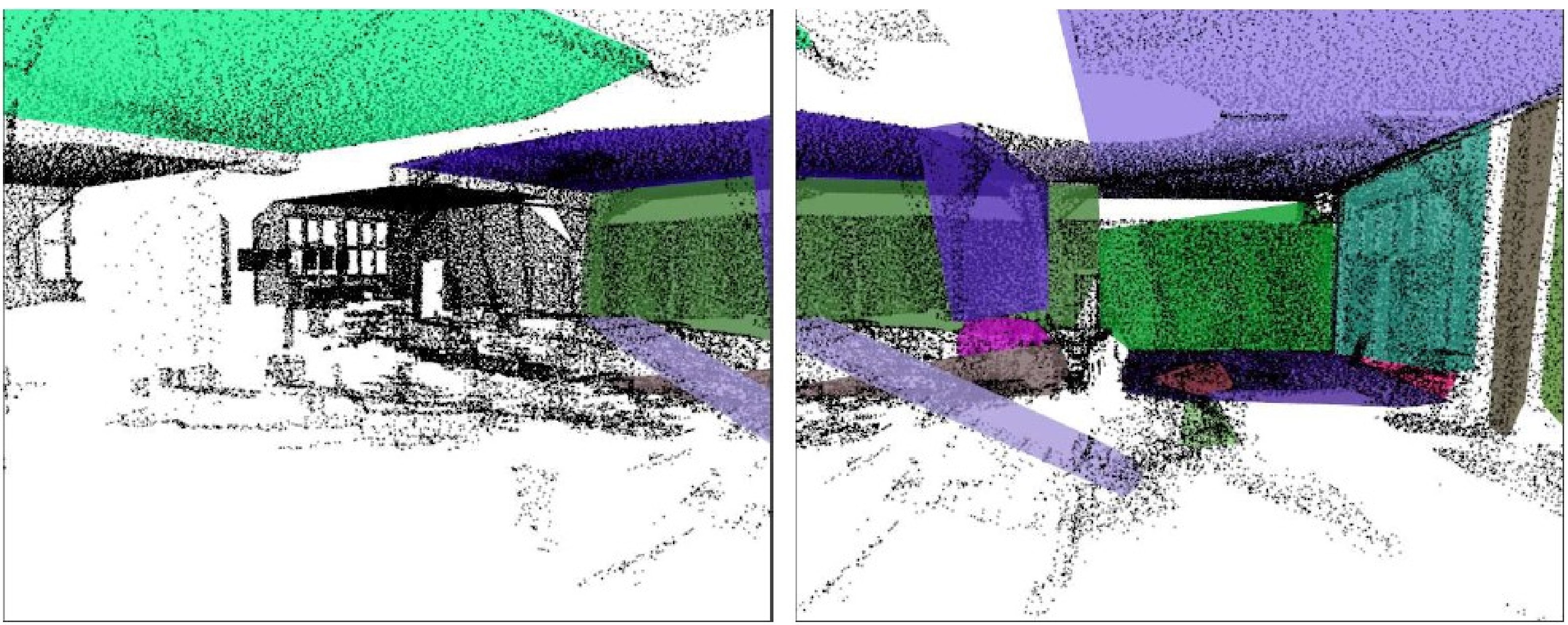


RG

Comparision with Related Work (3)



HFP



RG

Comparision with Related Work (4)

	# points	RHT	N	RHT C	N	RG	N	HFP
empty room	325, 171	0.072 +0.599 = 0.671	5	0.088 +0.730 = 0.818	5	5.31 +2.33 = 7.64	$\gg 5$	78.4
empty room	81, 631	0.096 +0.195 = 0.291	5	0.092 +0.200 = 0.292	5	1.22 +0.5 = 1.72	9	14.2
simu-lated	81, 360	0.049 +0.182 = 0.231	5	0.055 +0.199 = 0.254	5	1.33 +0.49 = 1.82	8	18.7
hall	81, 360	2.813 +0.234 = 3.047	16	1.818 +0.277 = 2.095	17	1.35 +0.4 = 1.75	13	36.0
arena	144, 922	13.960 +0.477 = 13.960	18	6.930 +0.662 = 7.592	18	2.13 +0.57 = 2.70	11	16.0