Techniques for 3D Mapping and Object Recognition

W. Burgard

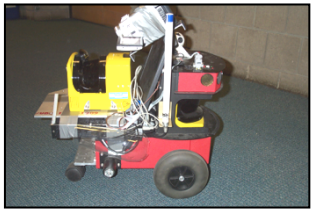
University of Freiburg

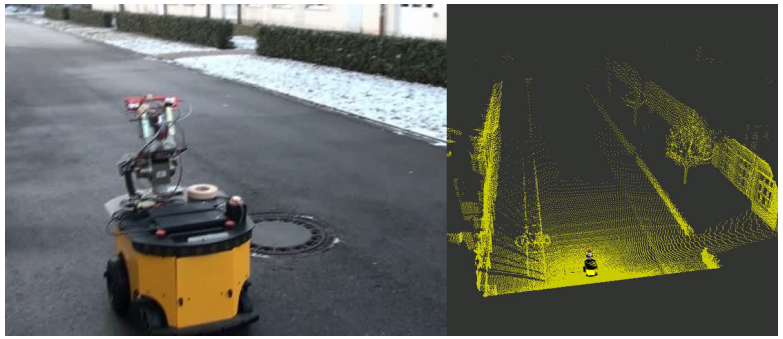
Why 3D?

- Robots operate in the three-dimensional world
- Three-dimensional maps support
  - object recognition
  - more accurate path planning
  - more reliable localization and data association
  - navigation on uneven terrain

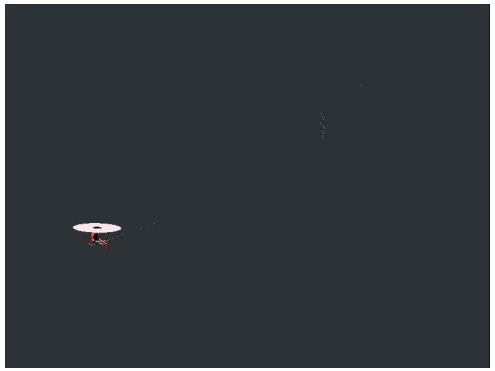
How to Generate 3D Data with Range Scanners?

- Combination of fixed scanners
- Mounted on manipulators
- Integrated scanners
- Rotating scanners
- Nodding units
- Free flying

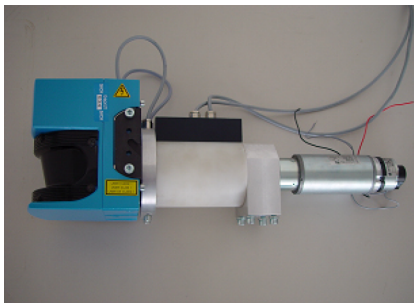




[courtesy A. Nüchter]



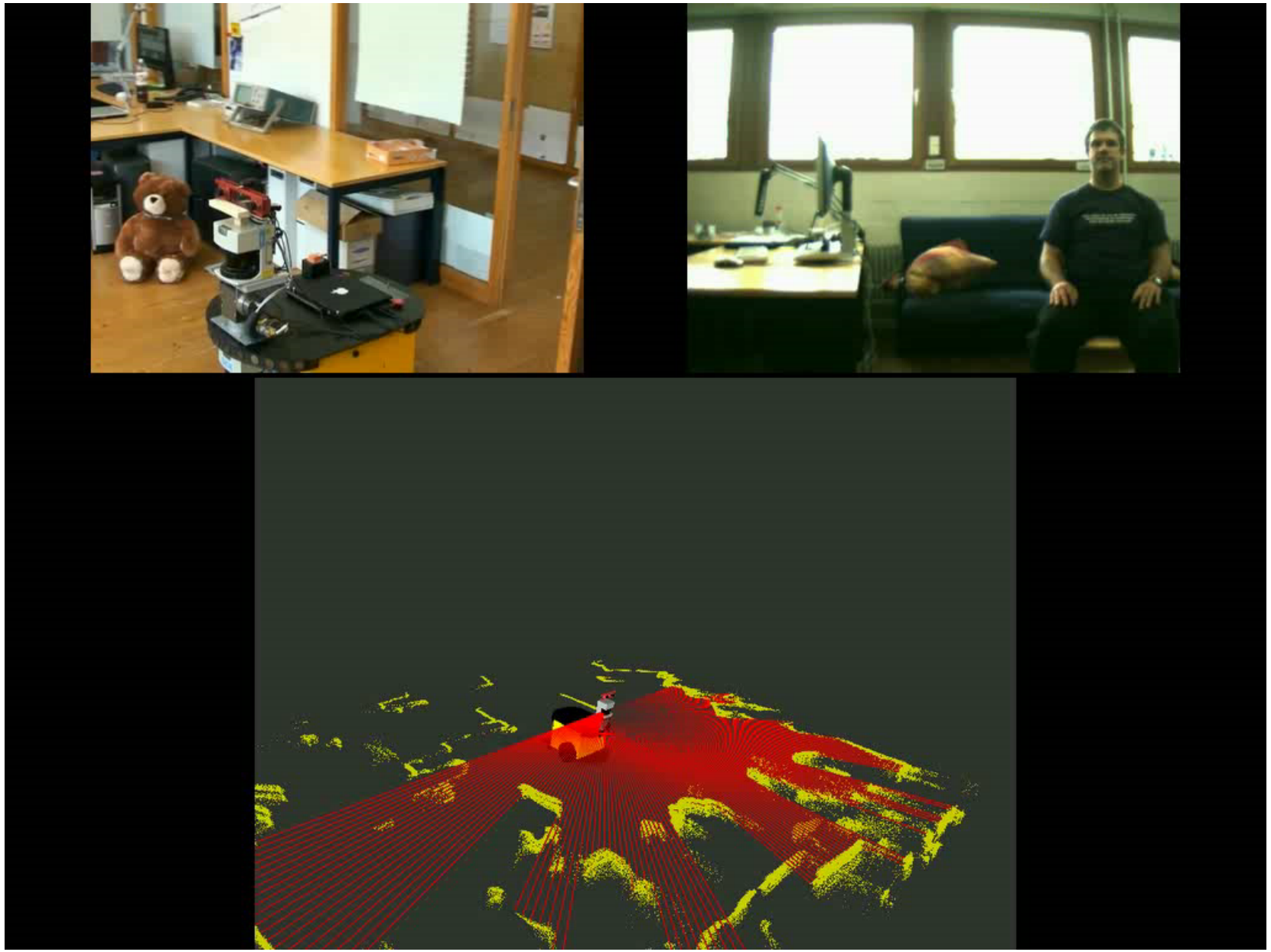
[courtesy D. Haehnel]



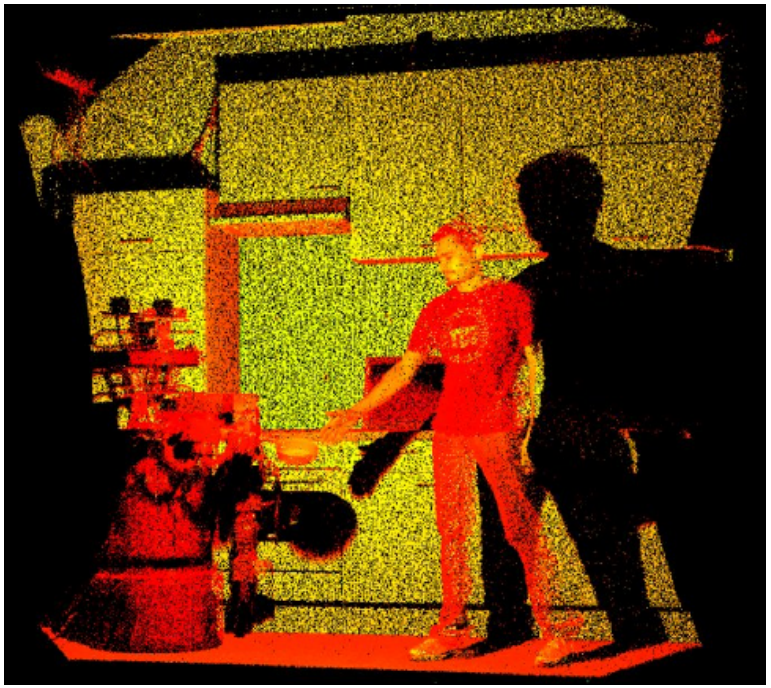
[courtesy P. Steinhaus]

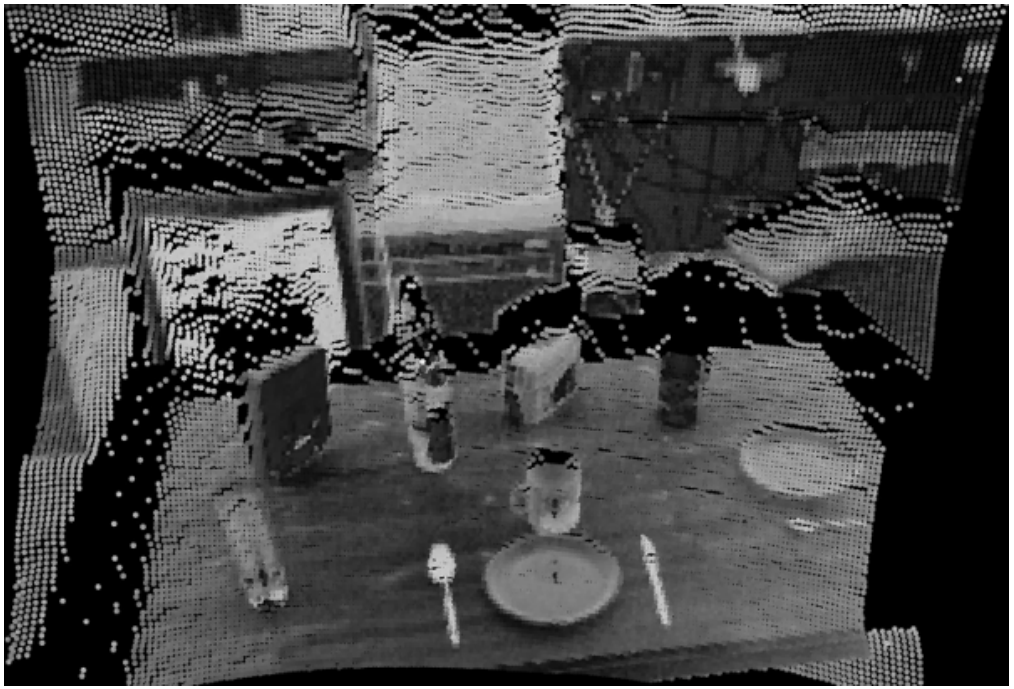
[courtesy D. Haehnel]

Acquisition Example



Point Clouds

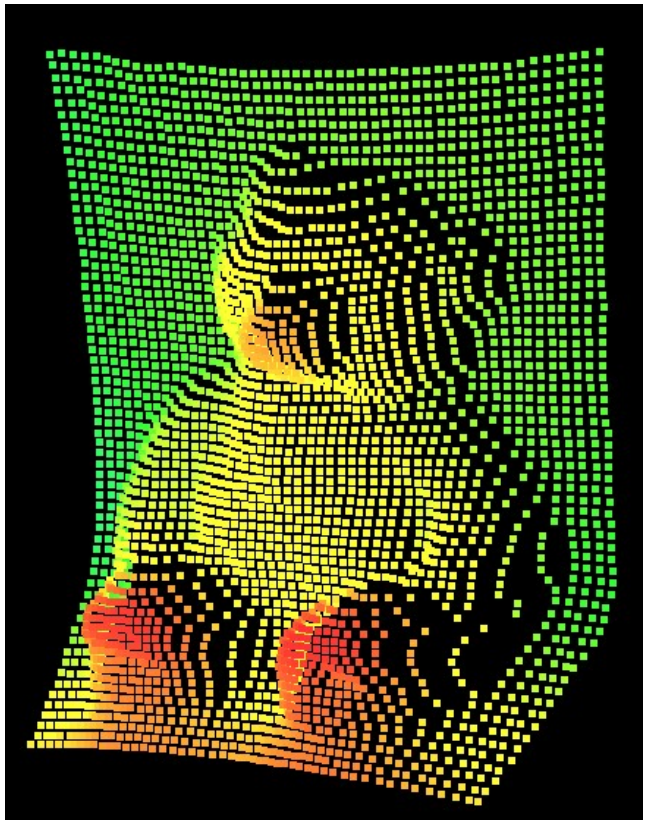




- Point Cloud = a "cloud" (i.e., collection) of nD points (usually n = 3)
- $\mathrm{p}_i = \{x_i, y_i, z_i\} \longrightarrow \mathcal{P} = \{\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_i, \ldots, \mathrm{p}_n\}$
- used to represent 3D information about the world

Alternative Source: Time of Flight Camera

- Pros: fast
- Cons: short distances only, indoors only



Sources: Stereo camera

- Pros: passive, no distance limitation, fast
- Cons: dependent on texture and light

The quality is distance dependent

Alternative Source: Kinect (Xbox360)

- Pros: cheap, fast
- Cons: only short distances, only indoor

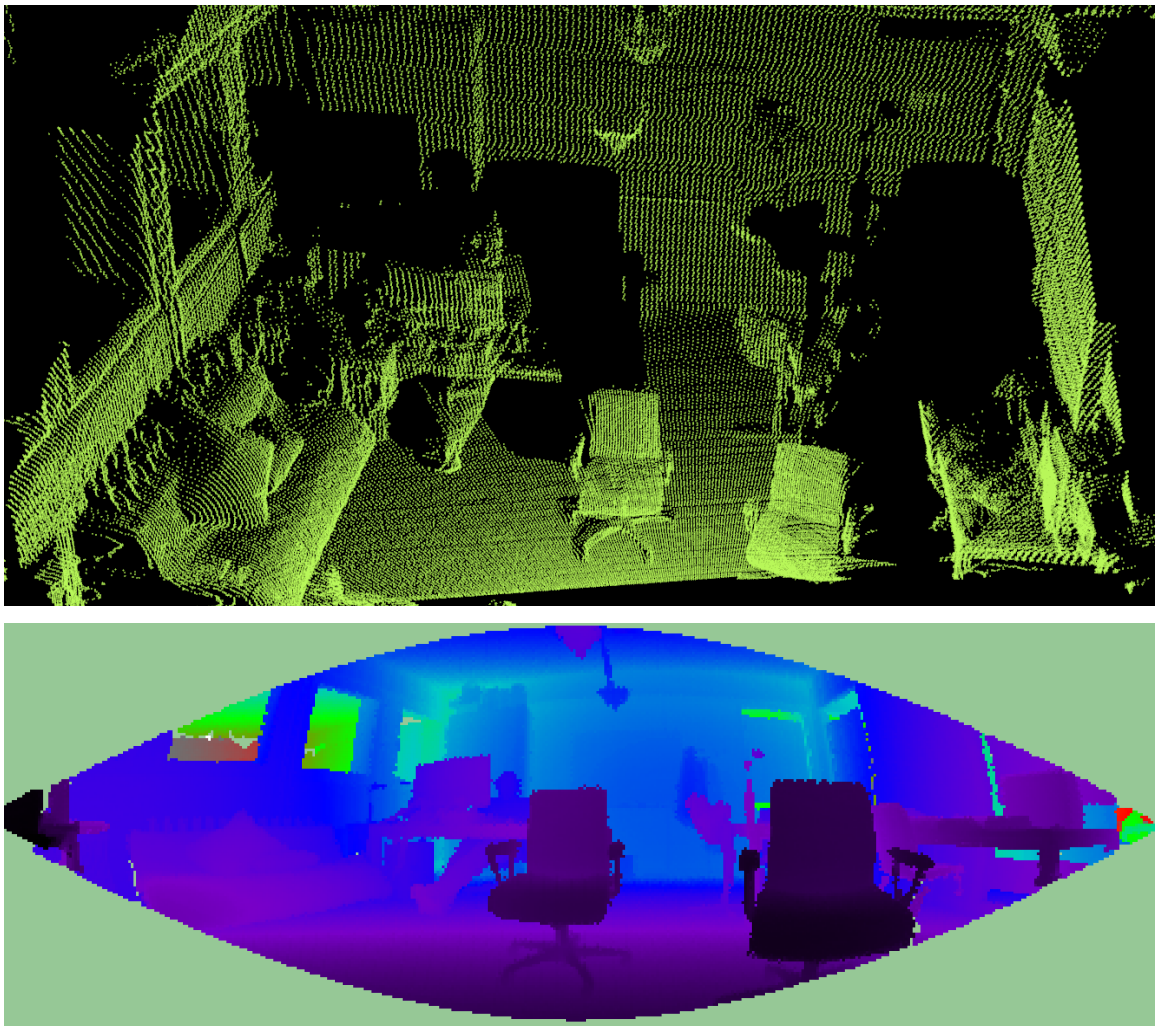
The quality is distance dependent

Example

Range Images

- Every pixel encodes the distance to the closest obstacle
- Dual representation to point clouds computed from one position

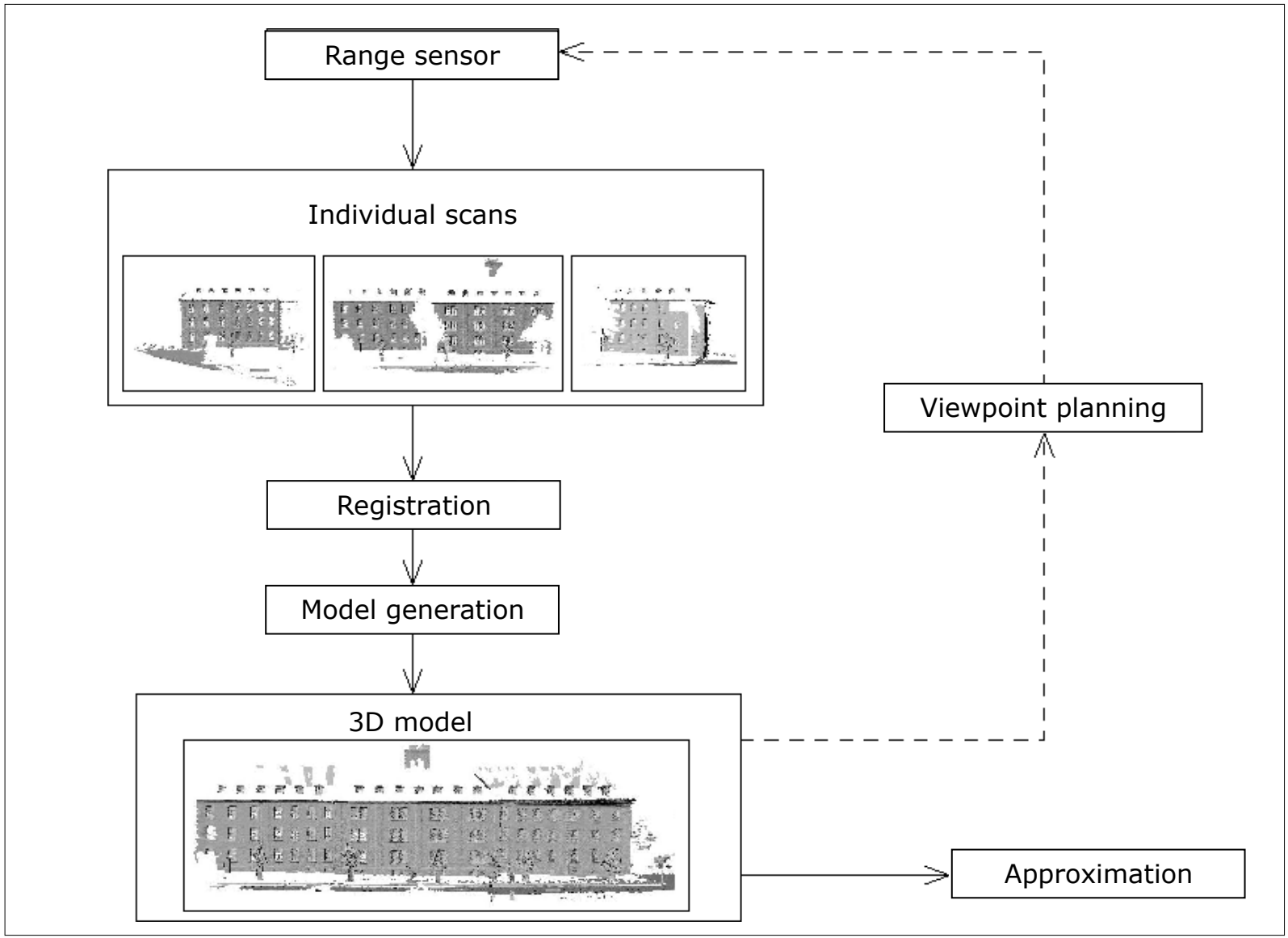
Point Clouds vs. Range Images



3D: Problems to be solved

- Data acquisition
- Viewpoint planning
- Scan registration
- Data fusion
- Approximation
- Navigation
- Search
- Object recognition
- …

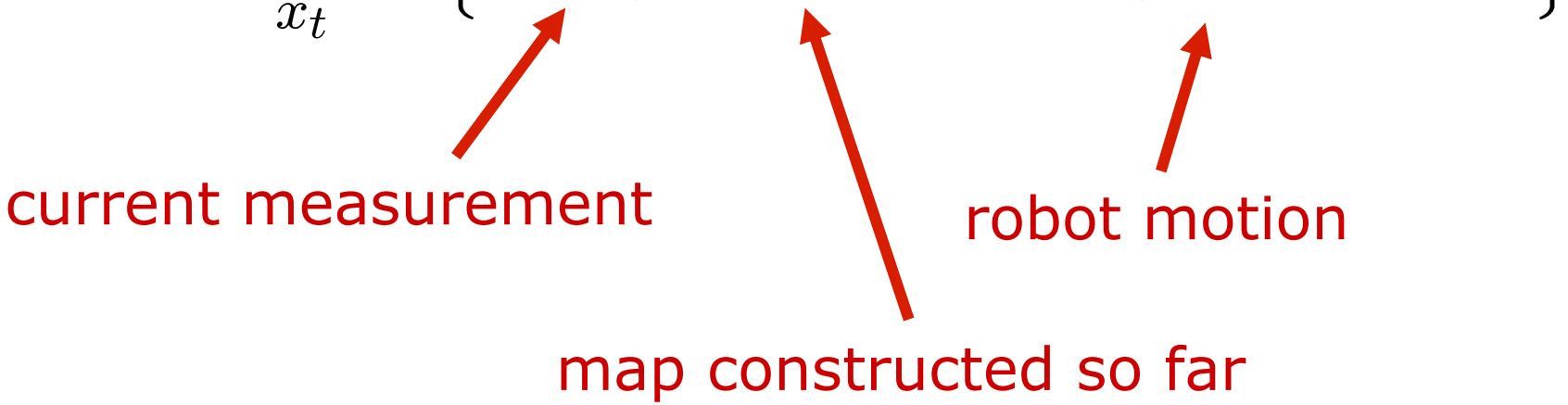
The 3D Mapping Process



Scan Matching

Maximize the likelihood of the i-th pose and observation relative to the (i-1)-th pose and map given the movement.

$$\hat{x}_t = \operatorname*{argmax}_{x_t} \left\{ p(z_t \mid x_t, \hat{m}^{[t]}) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \right\}$$



ICP

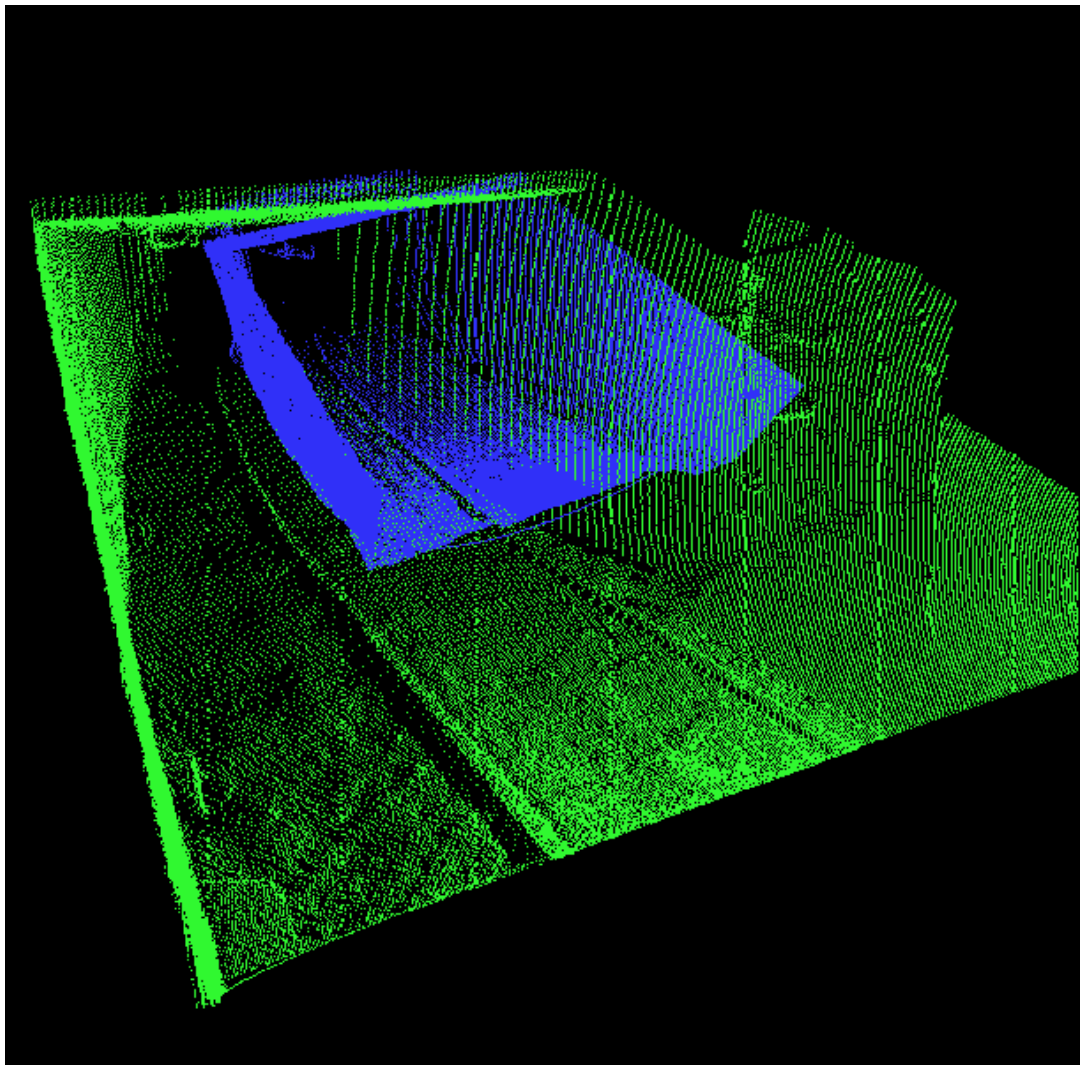
Maximize the likelihood of the i-th pose and observation relative to the given map.

$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}} \left\{ p(z_t \mid x_t, \hat{m}^{[t]}) \right\}$$

current measurement

map constructed so far

Application Example



[courtesy M. Hertzberg and A. Nuechter]

ICP doesn’t Solve the SLAM Problem

… but it can be used to create constraints for Graph-SLAM techniques.

Techniques for Generating Consistent Maps

- Incremental SLAM
  - Gaussian Filter SLAM
    - Smith & Cheesman, ‘92
    - Castelanos et al., ‘99
    - Dissanayake et al., ‘01
    - …
  - Fast-SLAM
    - Murphy et al., ‘99
    - Montemerlo et al., ‘02/‘03
    - Haehnel, ‘03
    - Grisetti et al., ‘04
    - …
- Full SLAM
  - EM
    - Thrun et al., ‘98
    - Burgard et al., ‘99
  - **Graph-SLAM**
    - Folkesson et al., ‘98
    - Frese et al., ‘03
    - Howard et al., ‘05
    - Dellaert et al., ‘05
    - Thrun et al., ‘05
    - Olson, ‘06
    - Grisetti et al., ‘08

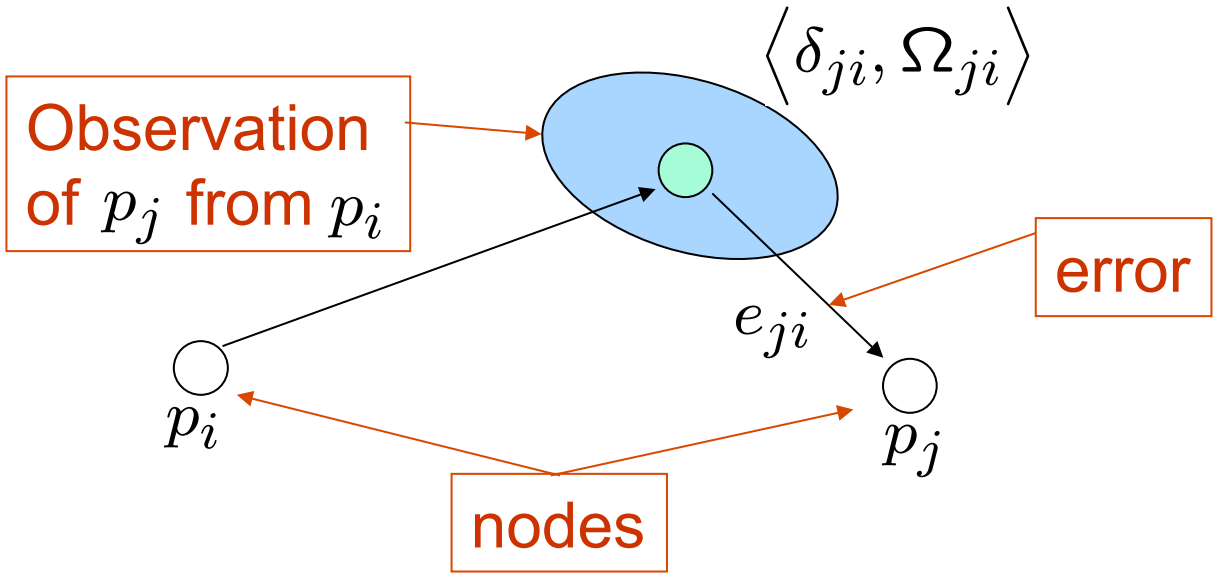
Graph-based SLAM

- Use a **graph** to represent the problem
- **Every node** in the graph **corresponds to a pose** of the robot during mapping
- **Every edge** between two nodes **corresponds to the spatial constraints** between them

- **Goal**: Find a configuration of the nodes that **minimize the error** introduced by the constraints

Problem Formulation

- The problem can be described by a graph

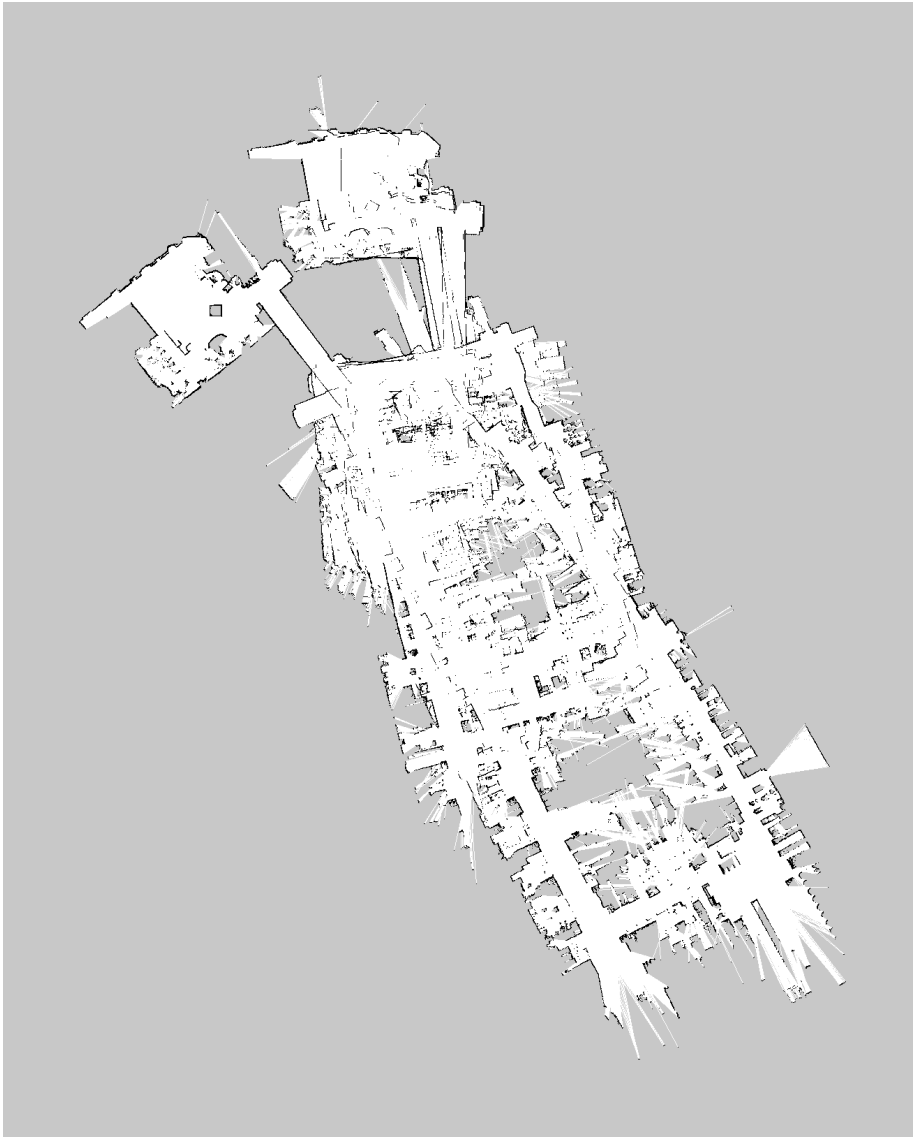


Goal: Find the assignment of poses to the nodes of the graph which minimizes the negative log likelihood of the observations:

$$\mathbf{p}^* = \text{argmin} \sum_{ji} e_{ji}^T \Omega'_{ji} e_{ji}$$

Graph-Based SLAM in a Nutshell

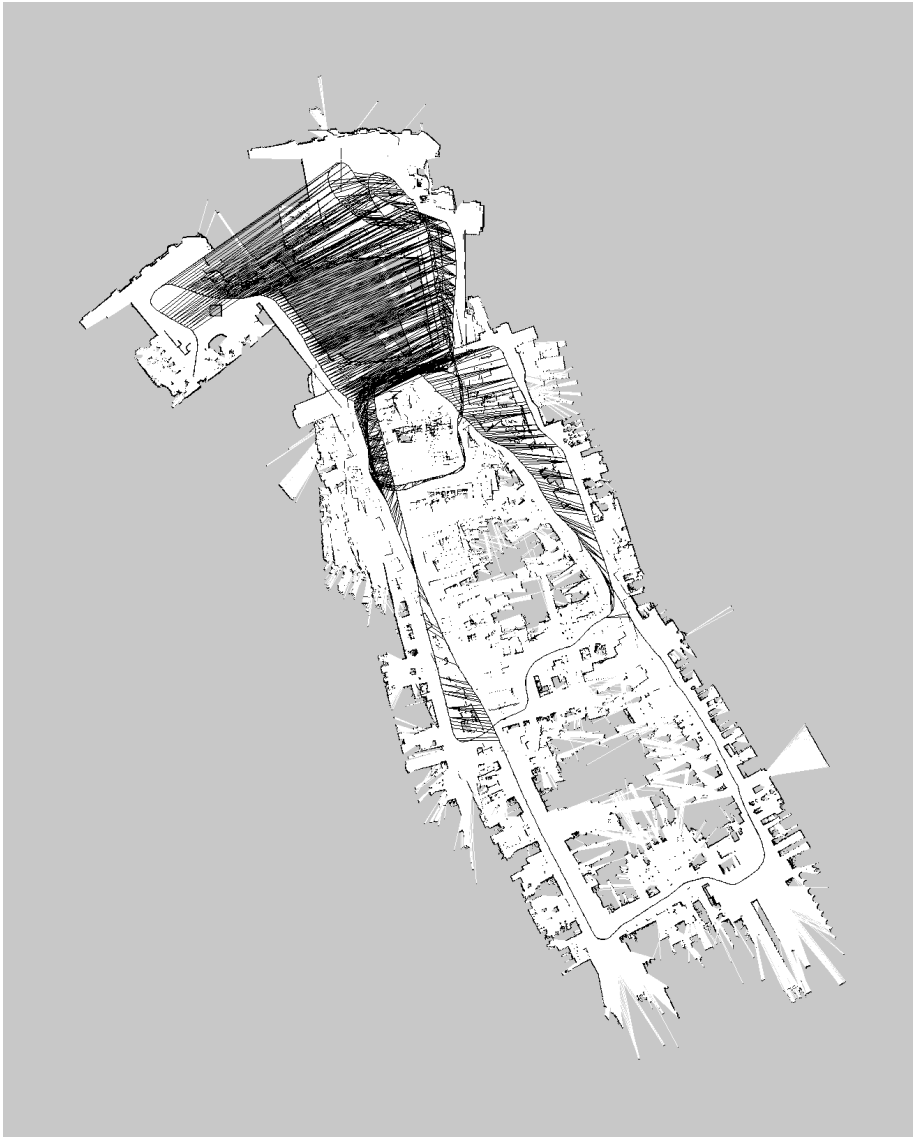
- Problem described as a graph
  - Every node corresponds to a robot position and to a laser measurement
  - An edge between two nodes represents a data-dependent spatial constraint between the nodes



[KUKA Hall 22, courtesy P. Pfaff]

Graph-Based SLAM in a Nutshell

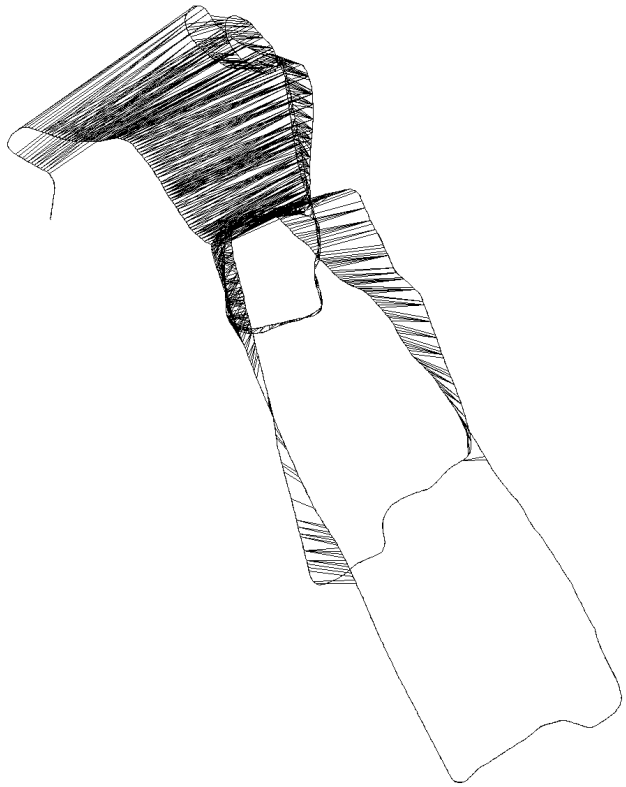
- Problem described as a graph
  - Every node corresponds to a robot position and to a laser measurement
  - An edge between two nodes represents a data-dependent spatial constraint between the nodes



[KUKA Hall 22, courtesy P. Pfaff]

Graph-Based SLAM in a Nutshell

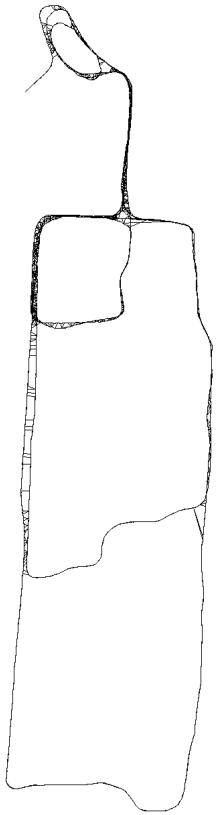
- Once we have the graph, we determine the most likely map by "moving" the nodes



[KUKA Hall 22, courtesy P. Pfaff]

Graph-Based SLAM in a Nutshell

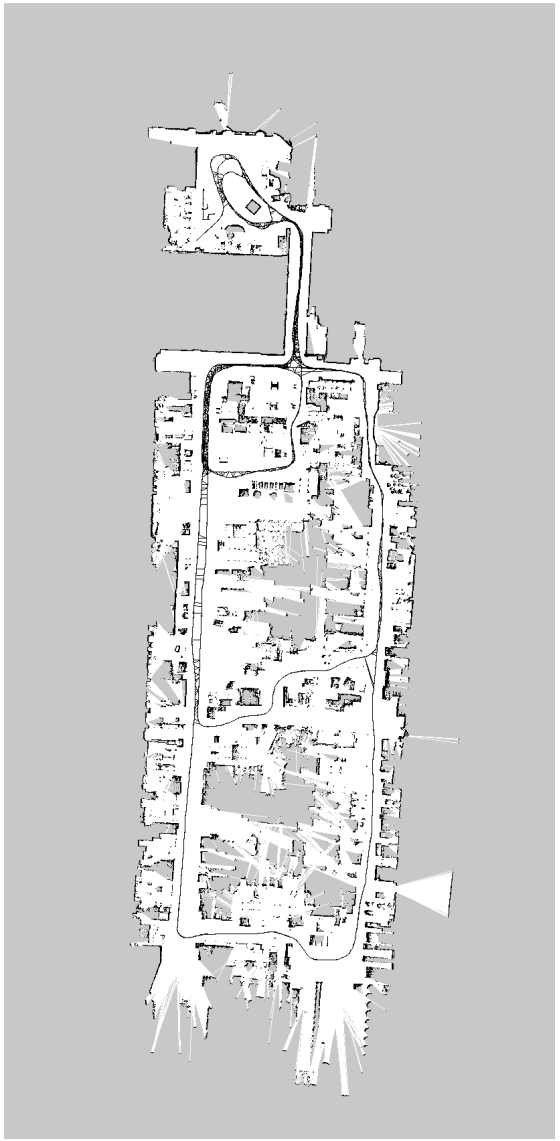
- Once we have the graph, we determine the most likely map by "moving" the nodes
- … like this.



[KUKA Hall 22, courtesy P. Pfaff]

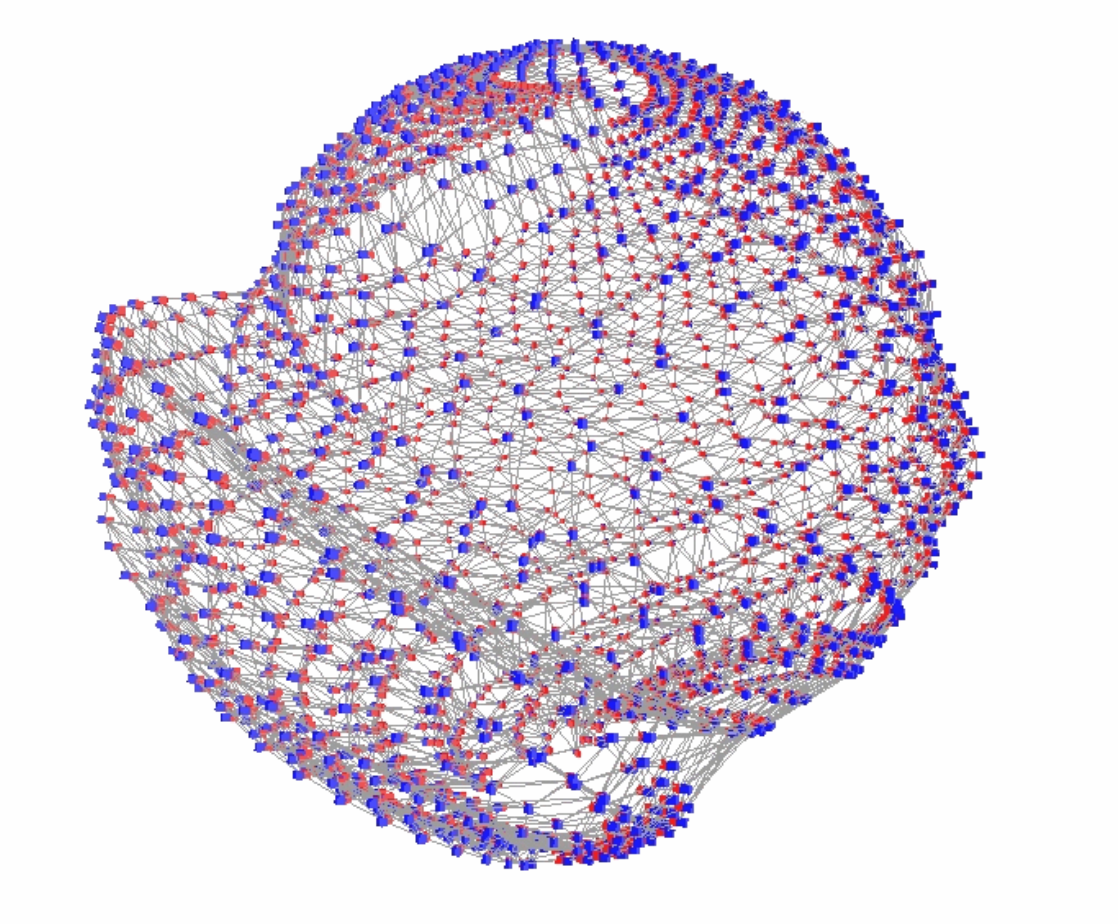
Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by "moving" the nodes
- … like this.
- Then we render a map based on the known poses

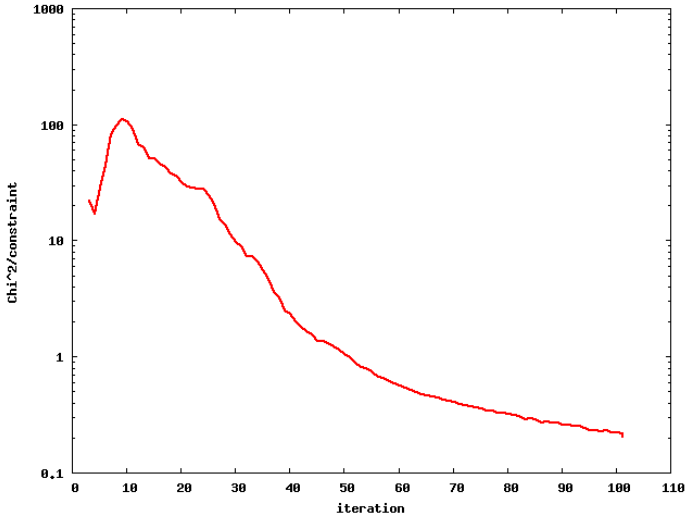


[KUKA Hall 22, courtesy G. Grisetti]

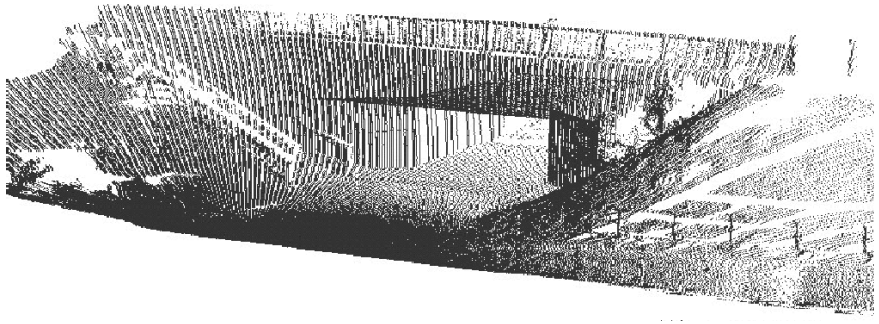
3D: Simulated Experiment



- Highly connected graph
- Poor initial guess
- LU & variants fail
- 2200 nodes
- 8600 constraints



Terrain Maps



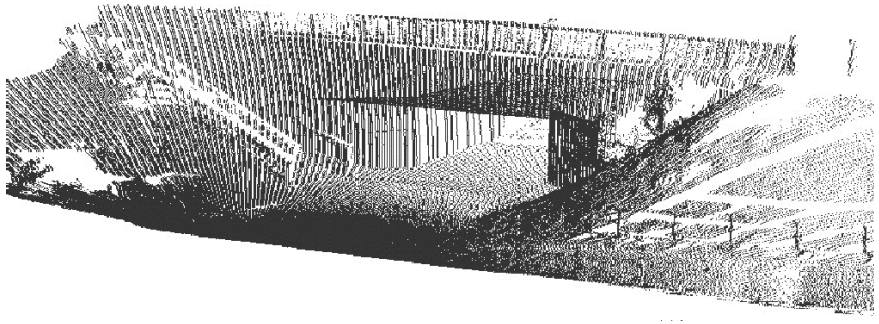
Point cloud

\+ Highly accurate representation

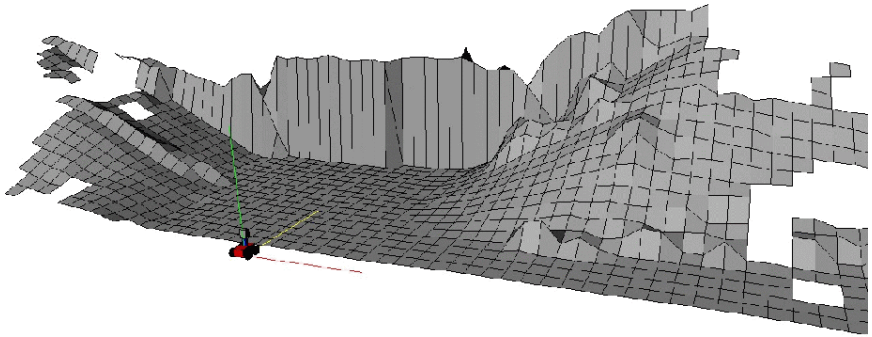
– Planning/navigation hard

– Huge memory requirements

Terrain Maps



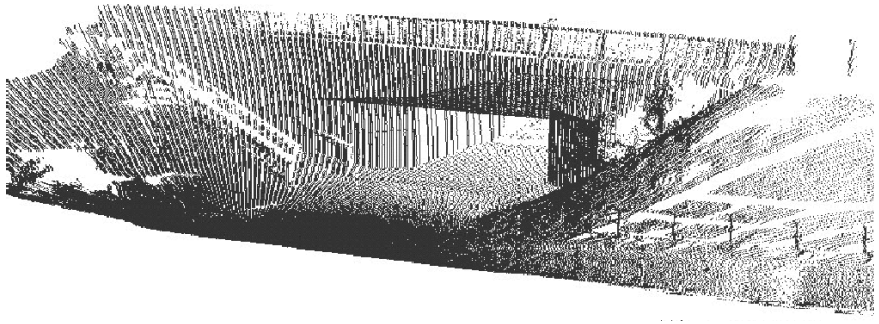
Point cloud



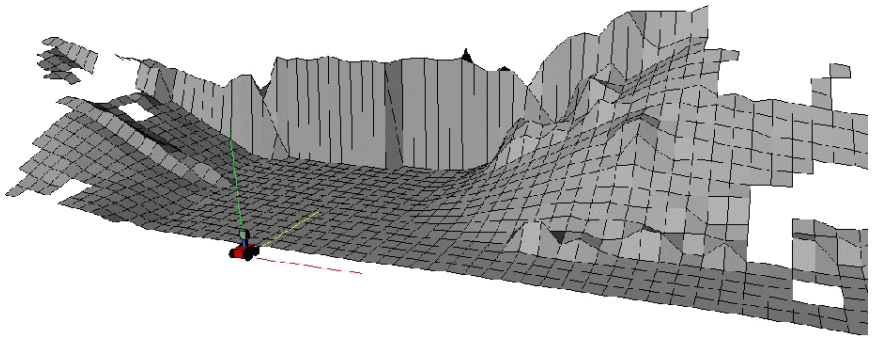
Standard elevation map

+ Planning and navigation possible (2.5D grid structure)
+ Compact representation
− Bridges appear as big obstacles

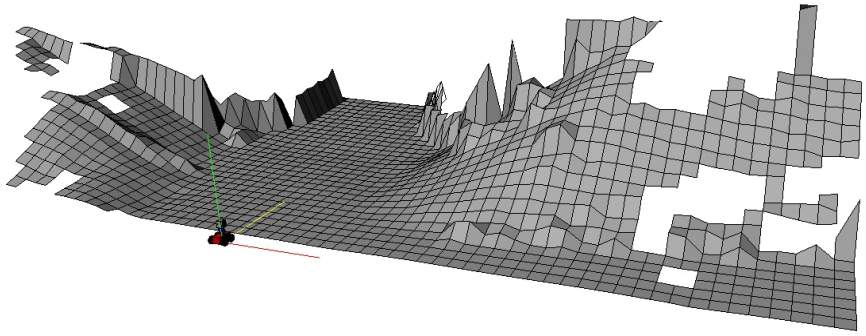
Terrain Maps



Point Cloud



Standard Elevation Map

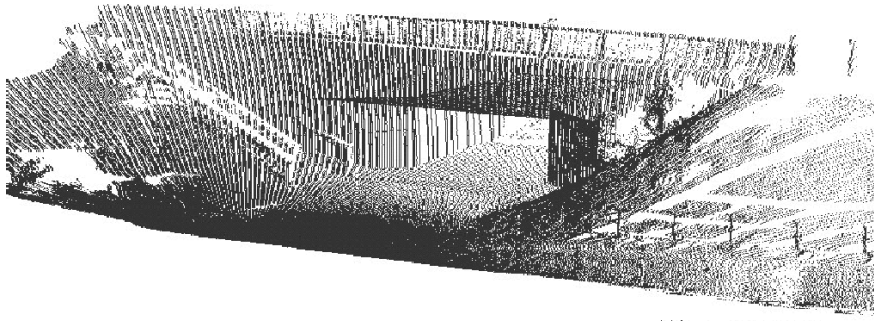


Extended elevation map

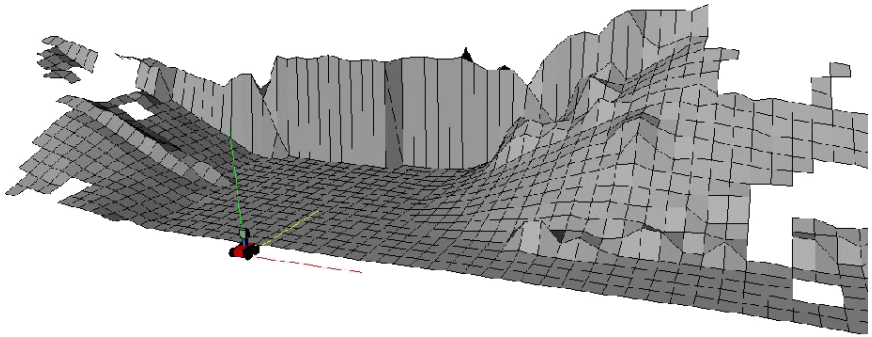
+ Planning with underpasses possible (cells with vertical gaps)

– No paths *passing under* and *crossing over* bridges possible (only one level per grid cell)

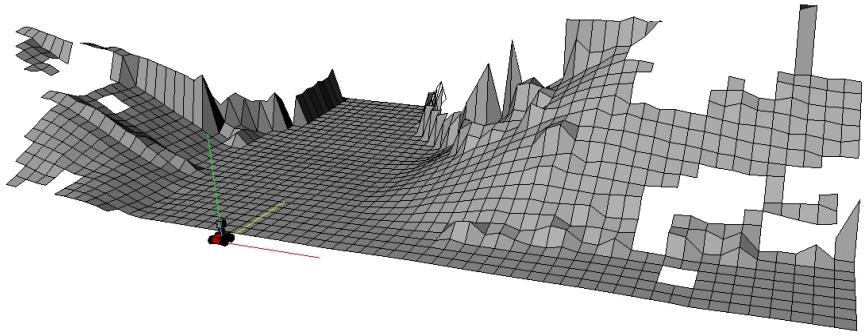
Terrain Maps



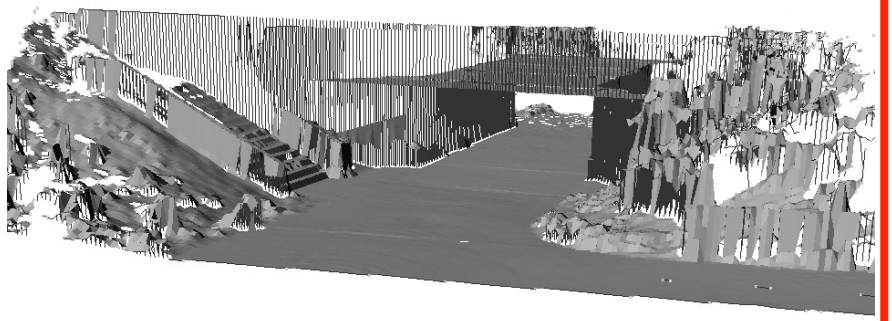
Point cloud



Standard elevation map

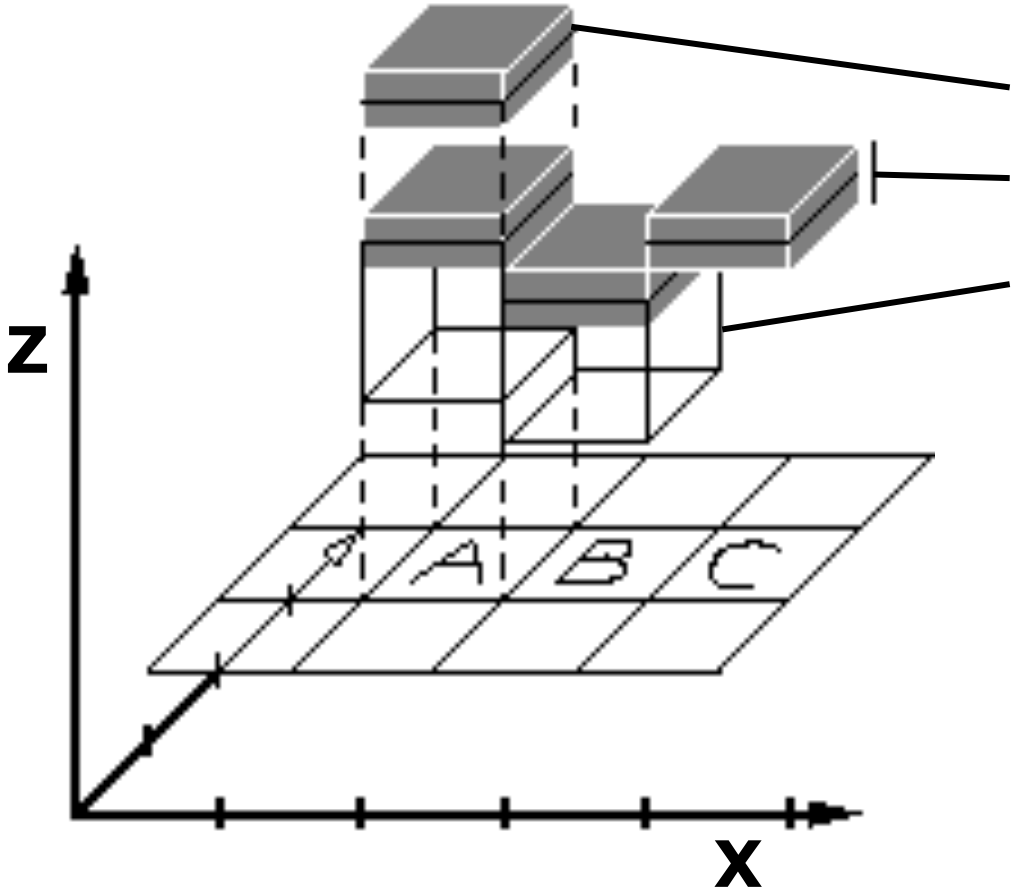


Extended elevation map



Multi-level surface map

MLS Map Representation



Each 2D *cell* stores various *patches* consisting of:

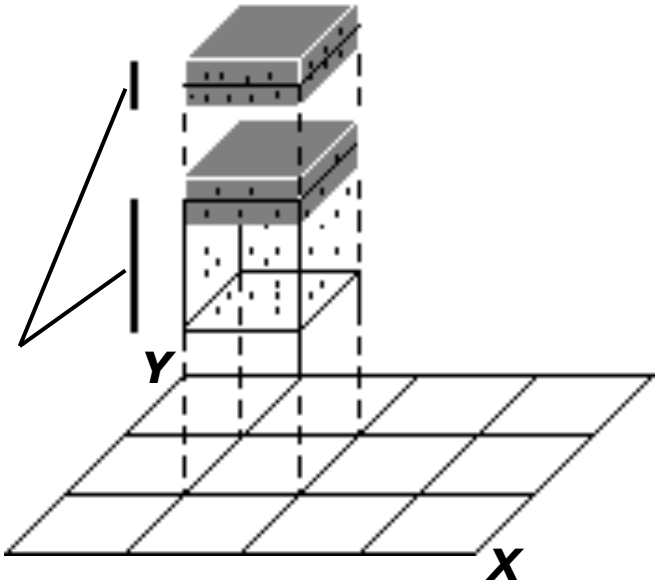
- A height mean μ
- A height variance σ
- A depth value d

- A *patch* can have no depth (flat objects, e.g., floor)
- A *cell* can have several patches (vertical gap cells, e.g., bridges)

From Point Clouds to MLS Maps

Map creation:

- Determine x,y cell for each point
- Compute vertical *intervals*
- Classify into *vertical* and *horizontal* intervals



- Apply Kalman update rule to all measurements in horizontal intervals (patches)
- Use highest measurement as mean in vertical intervals

Local Map Matching

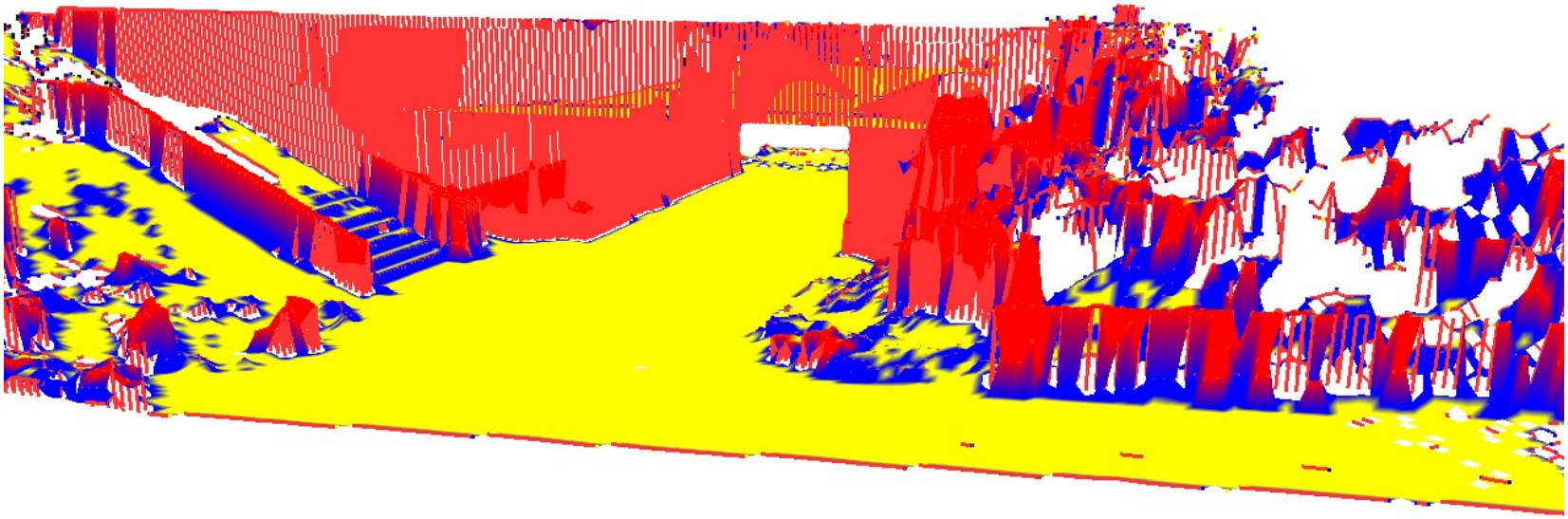
- Use Iterative Closest Point (ICP) algorithm
- Use surface patches instead of scan points
- Extract 3 classes of features from each map:
  - *vertical patches*
  - *horizontal and traversable patches*
  - *horizontal and non-traversable patches*
- **Improved data association using features**
- Sub-sampling of feature classes to reduce complexity

Local Map Matching (Summary)

- Classify surface patches into feature classes
- Sub-sample each feature class
- Find rotation R and translation t that minimize the error function

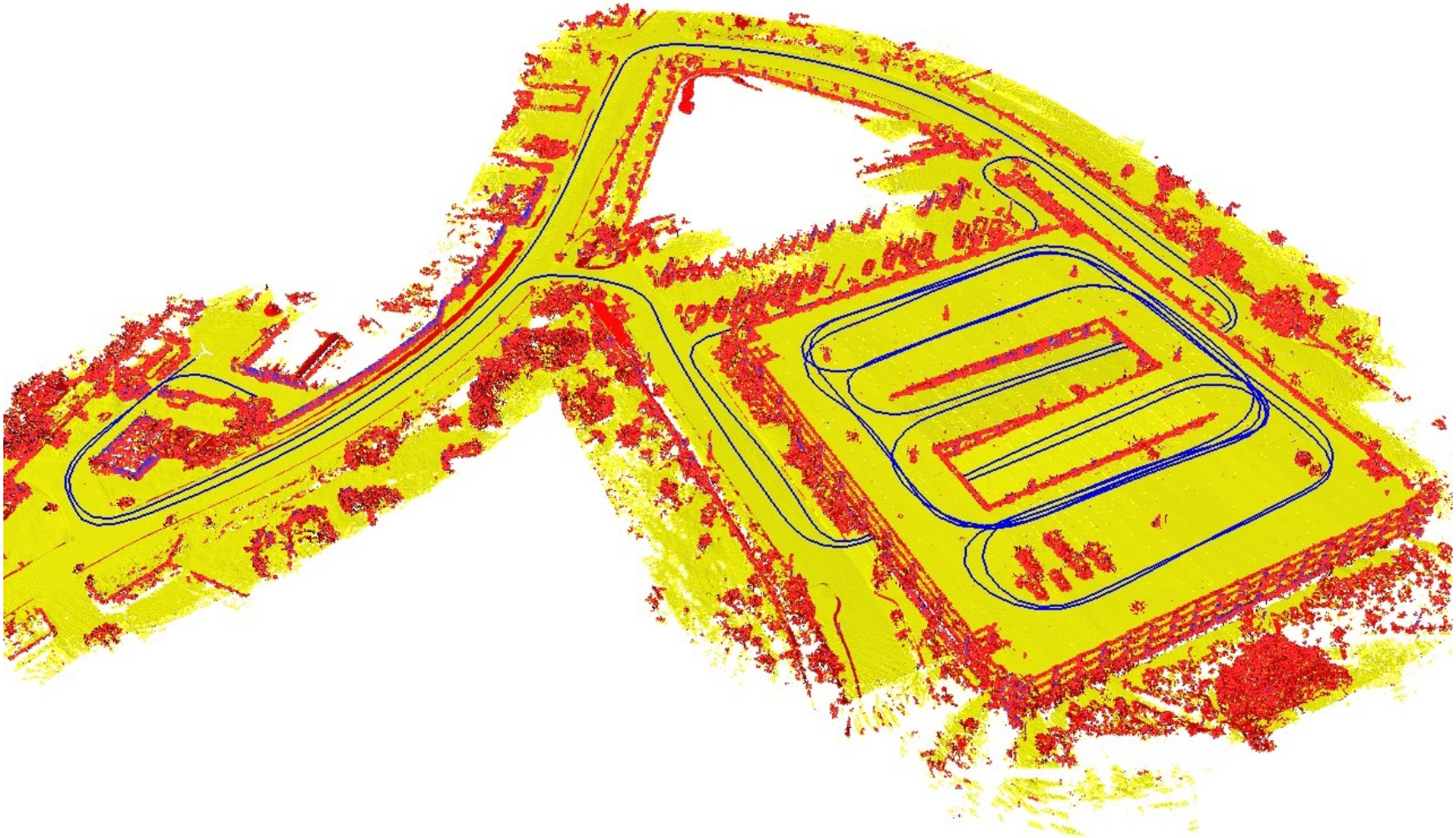
$$e(R,\mathbf{t}) = \underbrace{\sum_{c=1}^{C_1} d_v(\mathbf{u}_{i_c}, \mathbf{u}'_{j_c})}_{\text{vertical cells}} + \underbrace{\sum_{c=1}^{C_2} d(\mathbf{v}_{i_c}, \mathbf{v}'_{j_c})}_{\text{traversable}} + \underbrace{\sum_{c=1}^{C_3} d(\mathbf{w}_{i_c}, \mathbf{w}'_{j_c})}_{\text{non-traversable}}$$

Feature Extraction



- Traversable surface patches
- Cells with vertical objects
- Non-traversable surface patches

MLS Map of the Stanford Parking Garage

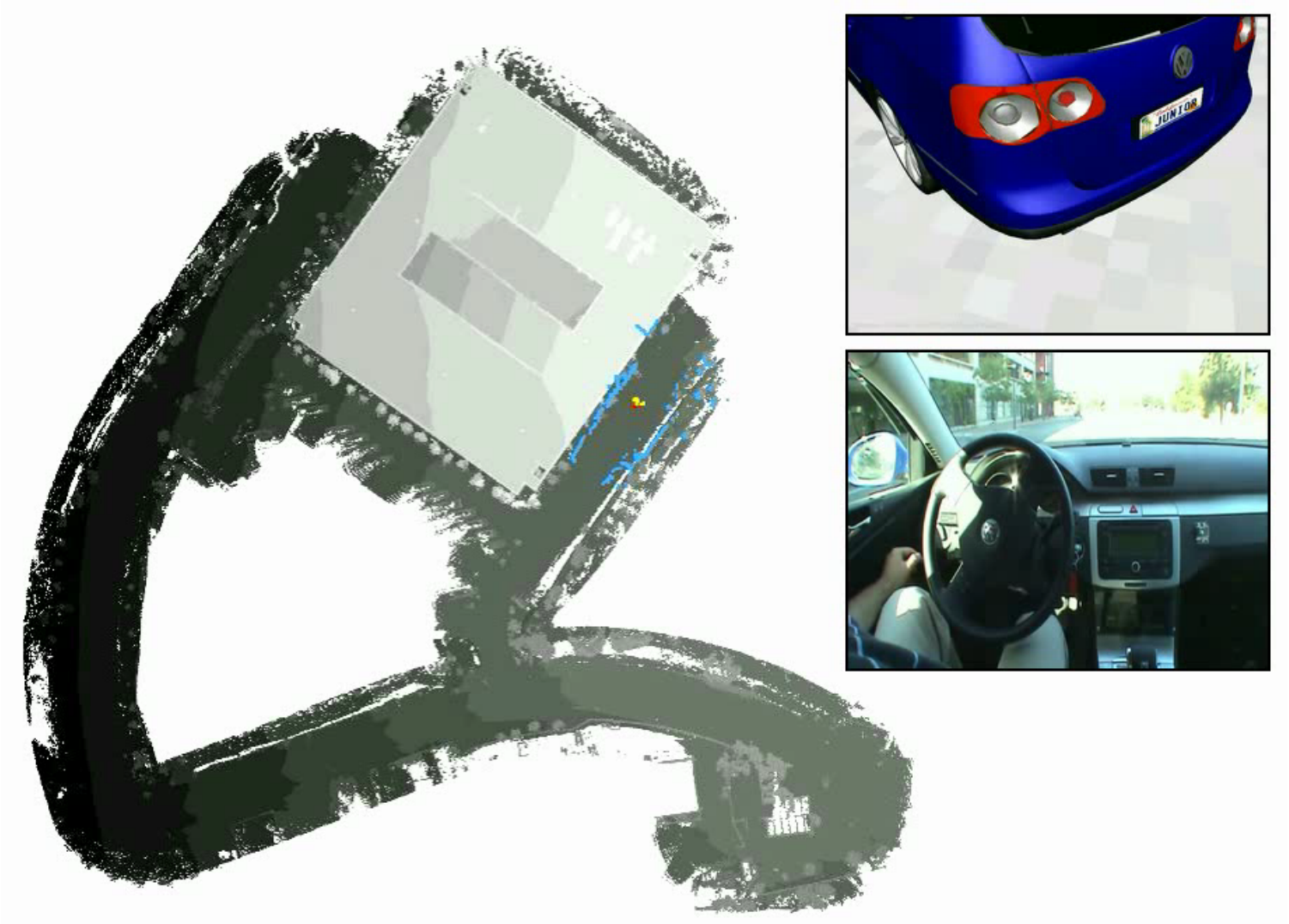


approx. 260MB

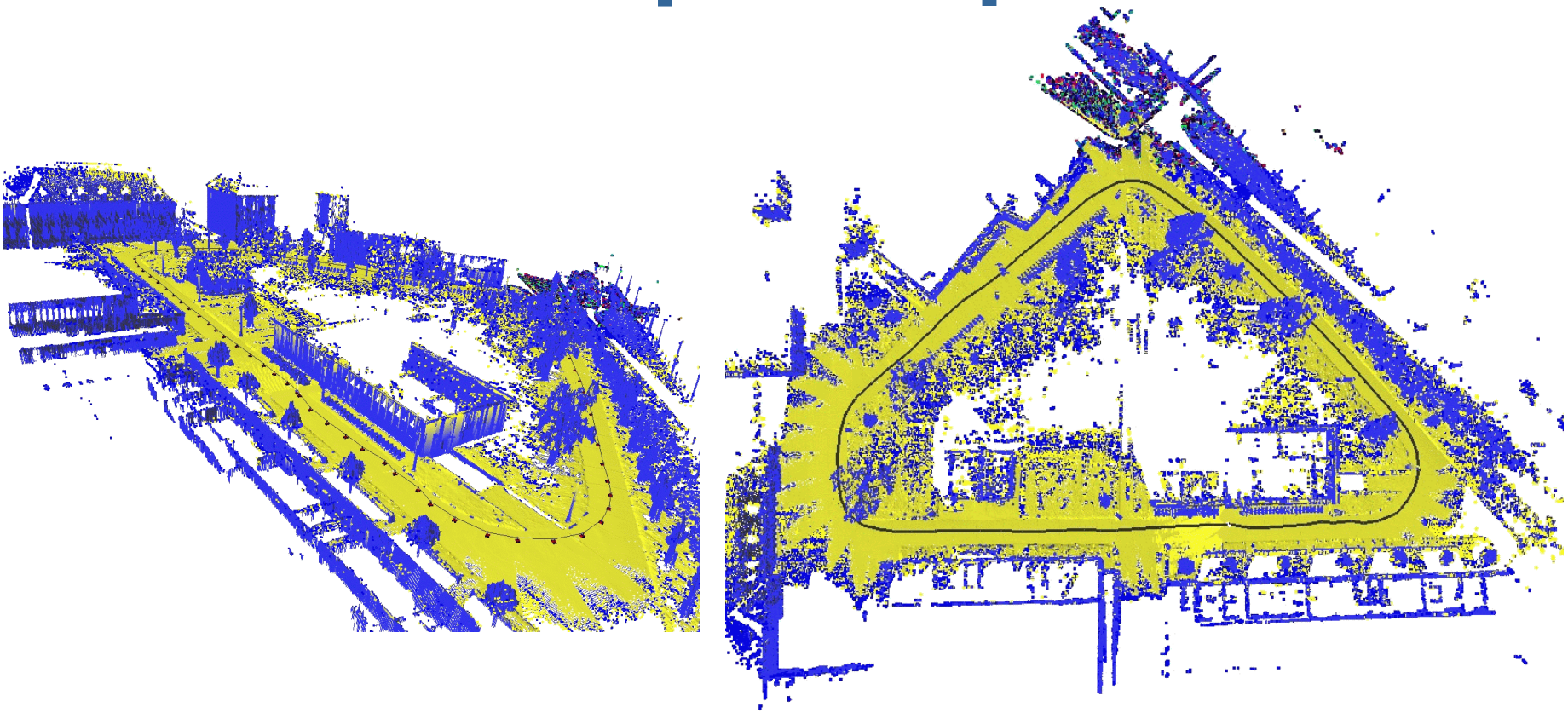
Navigation with the Autonomous Car Junior

- Task: reach a parking spot on the upper level of the garage.

Experiment with Junior



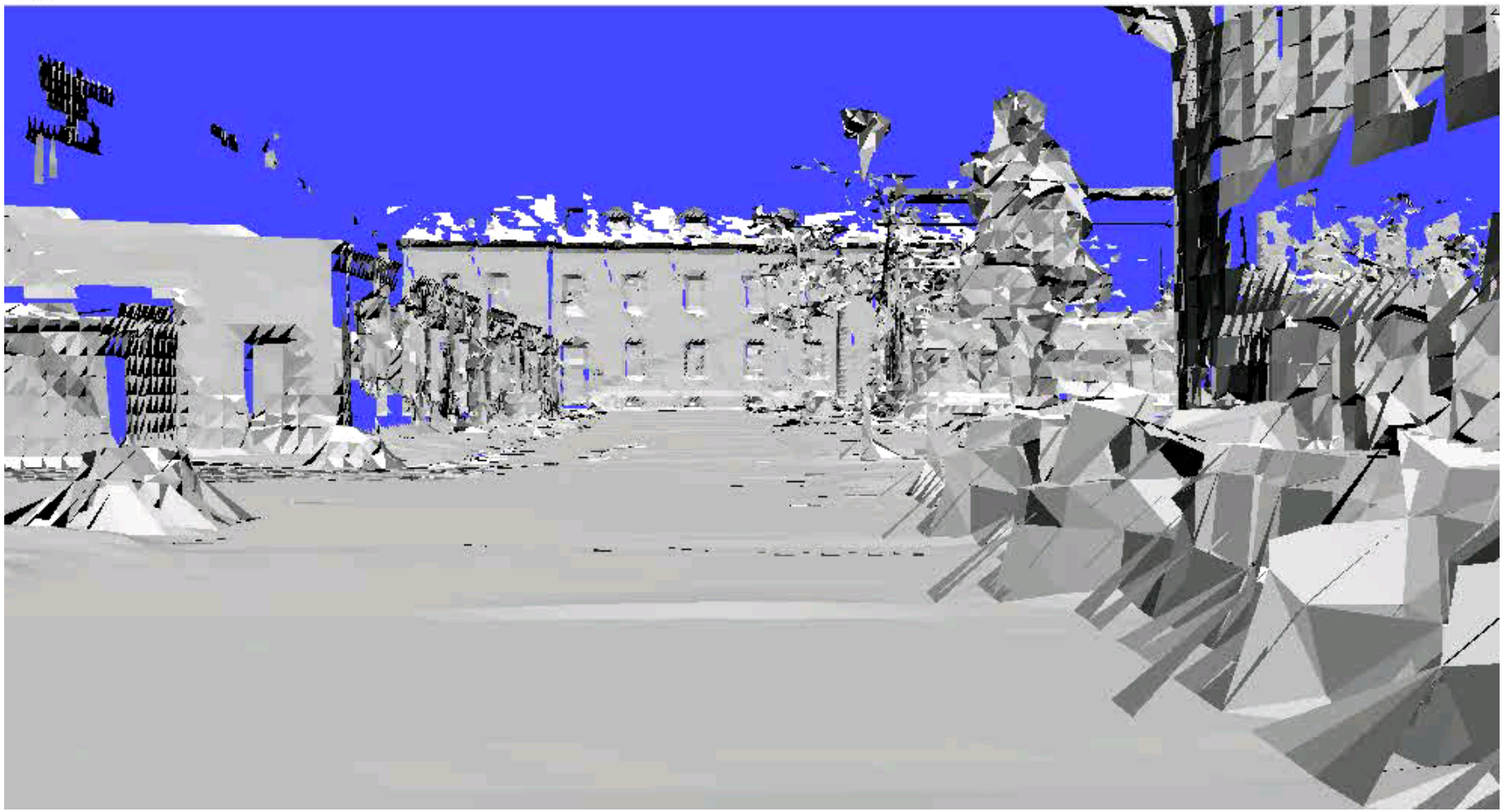
ALU-FR Campus Map



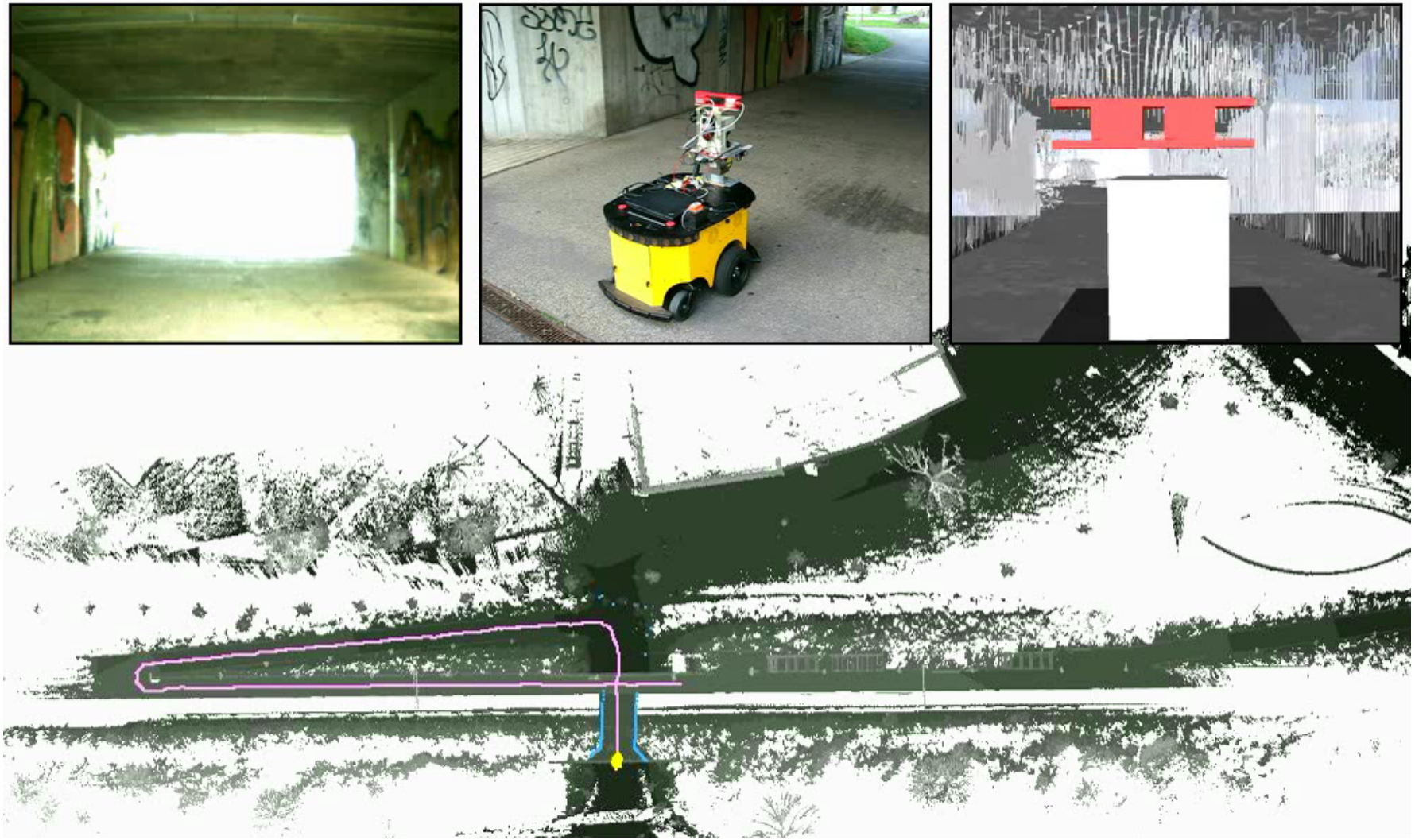
- Map size: 195 by 146 *m*
- Cell resolution: 10 *cm*
- Number of data points: 20,207,000

ALU-FR Campus Map

File

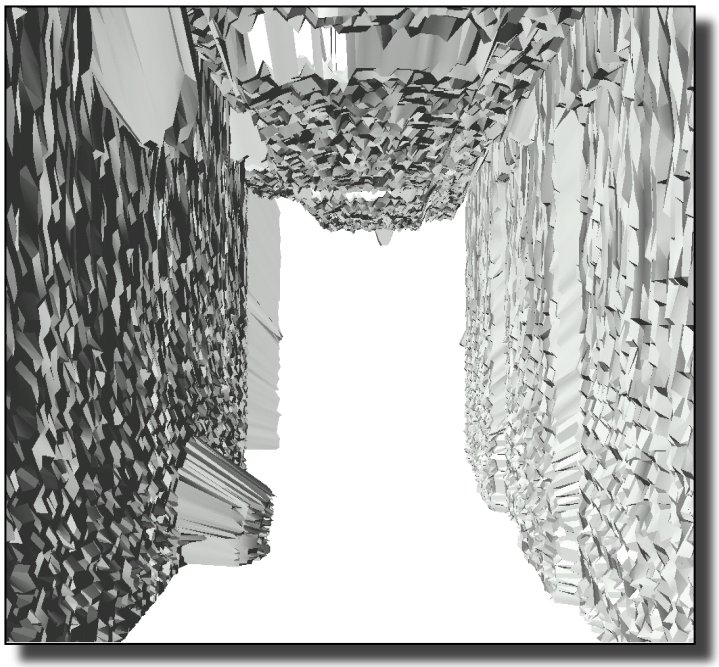


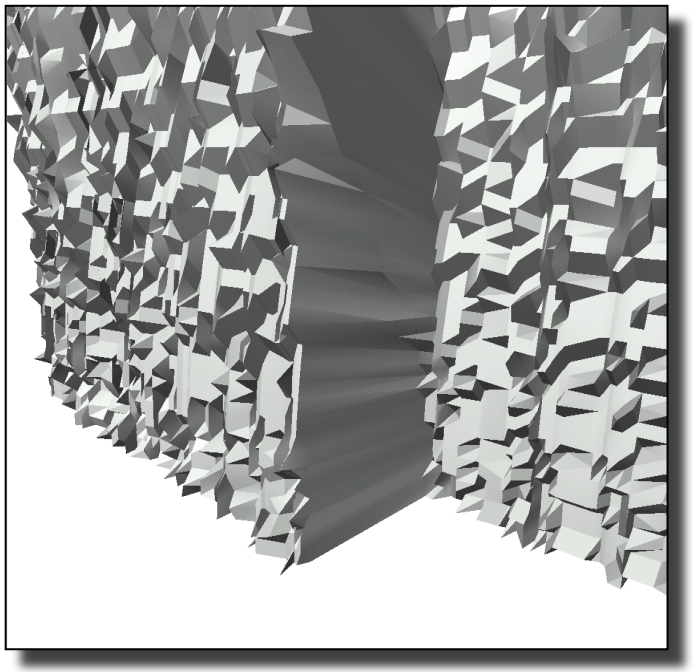
Planning with Multiple Levels



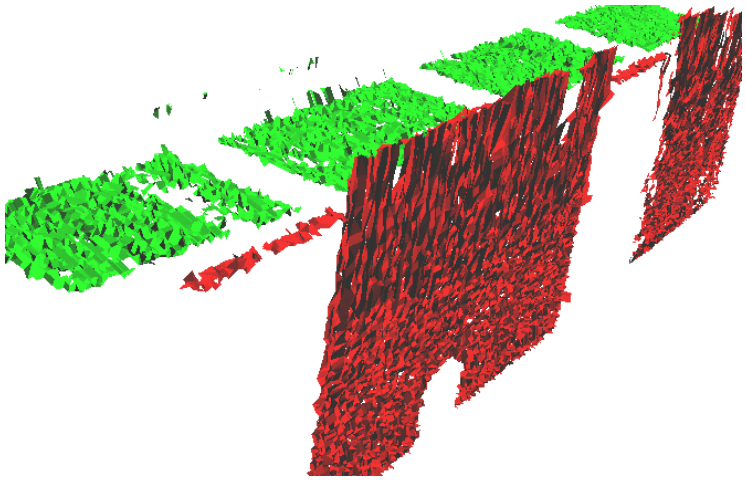
Why Planar Approximations?

- compactness
- accuracy

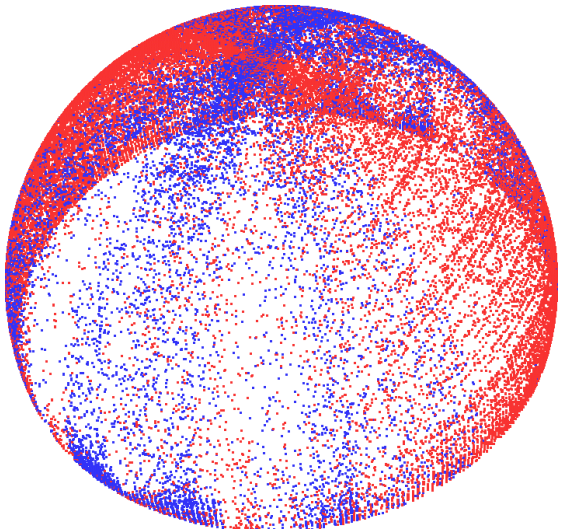




The Surface Normals



Partial data



Normals of triangles

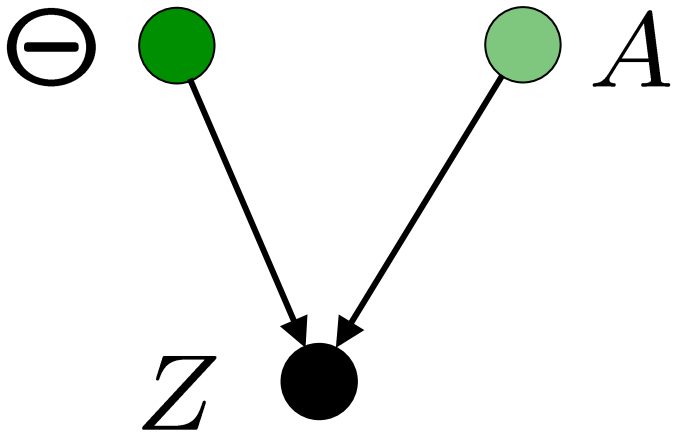
Plane Approximations

- Region growing
  [Besl and Jain ’88, …]
- Hough Transform
  [Okada et al. ’03, Sabe et al. ’04]
- Lines
  [Nuechter et al. ’03]
- Maximum Likelihood Clustering (EM)
  [Liu et al. ’03]

Plane Approximations

- Region growing
  [Besl and Jain ’88, …]
- Hough Transform
  [Okada et al. ’03, Sabe et al. ’04]
- Lines
  [Nuechter et al. ’03]
- Maximum Likelihood Clustering (EM)
  [Liu et al. ’03]

Graphical Model for Standard EM



Z	measurements
$\ominus$	planes
A	correspondences (hidden)

$$\text{JPD} = p(Z|A,\ominus)p(\ominus)p(A)$$

The E-Step in Standard EM

Calculate the expectations over the hidden variables:

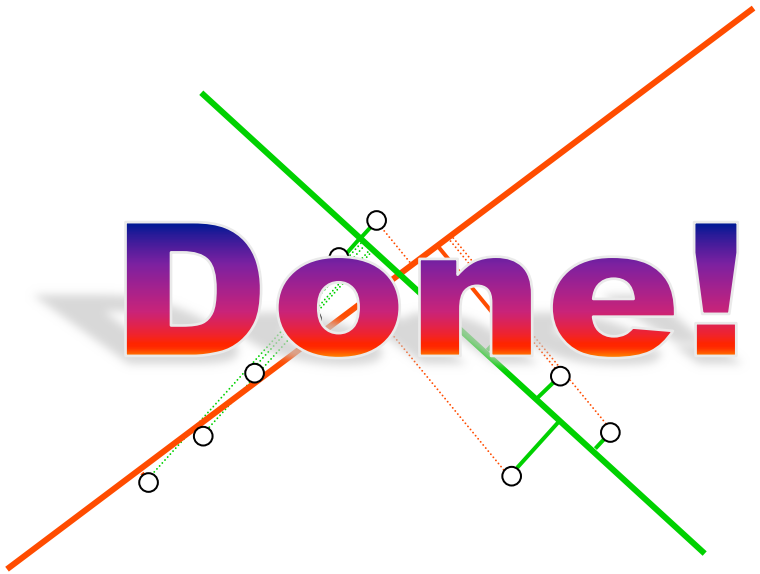
$$p(A, \Theta, Z) \propto \exp\left\{-\frac{1}{2}\sum_n \sum_m \alpha_{nm}\left(\frac{d_1(\mathbf{z}_n, \theta_m)}{\rho}\right)^2\right\}$$

$$E[\alpha_{nm} \mid \Theta] = \frac{\exp\left\{-\frac{1}{2}\left(\frac{d_1(\mathbf{z}_n,\theta_m)}{\rho}\right)^2\right\}}{\sum_j \exp\left\{-\frac{1}{2}\left(\frac{d_1(\mathbf{z}_n,\theta_j)}{\rho}\right)^2\right\}}$$

M-Step for Standard EM

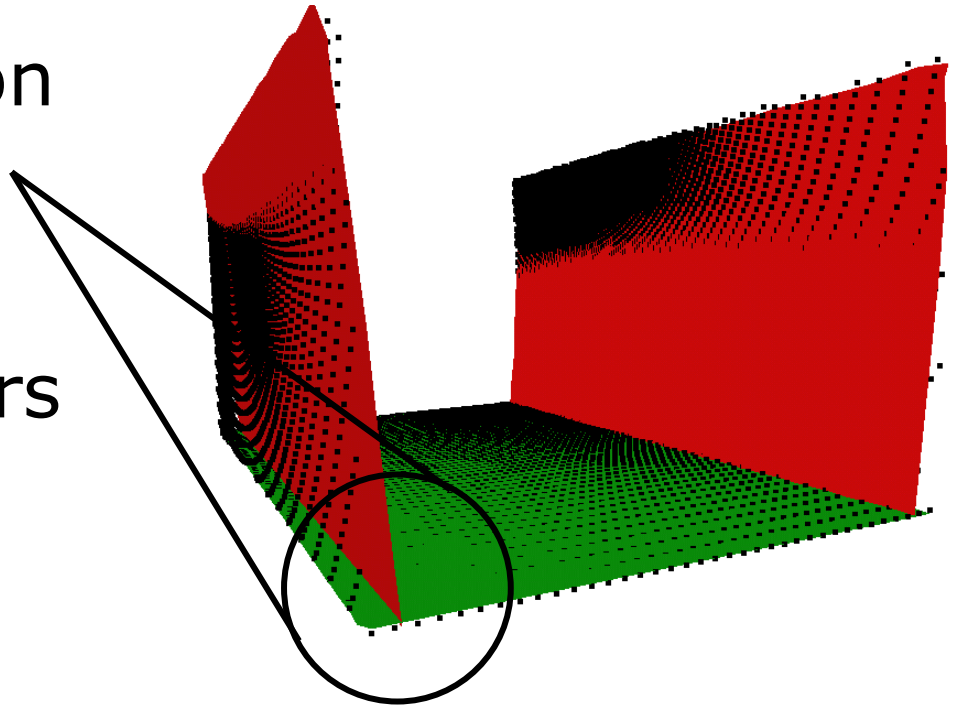
- Calculate the plane parameters
  - according to the data points
  - taking into account the estimated associations
- Can be done in closed form.

EM Example



Typical Problem in Standard EM

- Data association uncertainty
- results in alignment errors in M-step



Idea: Incorporate higher-level knowledge about planes (parallelism)

⇒ **Hierarchical EM**

Hierarchical Model

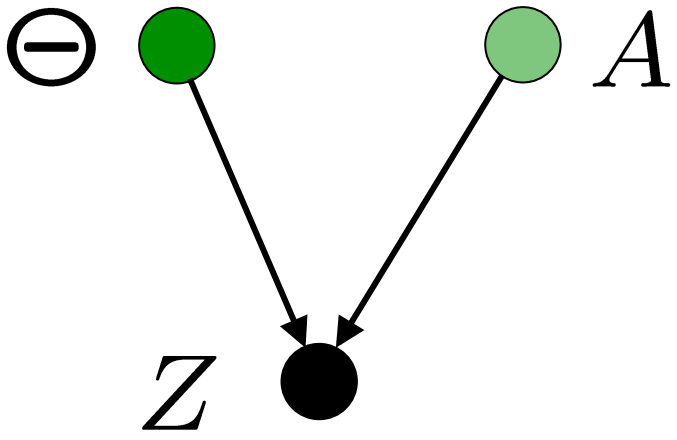
Idea: **simultaneously**

- cluster **points into planes** and
- **planes into main directions** to

- obtain the *most likely* set of planes and main directions.

Maximum Likelihood Estimation

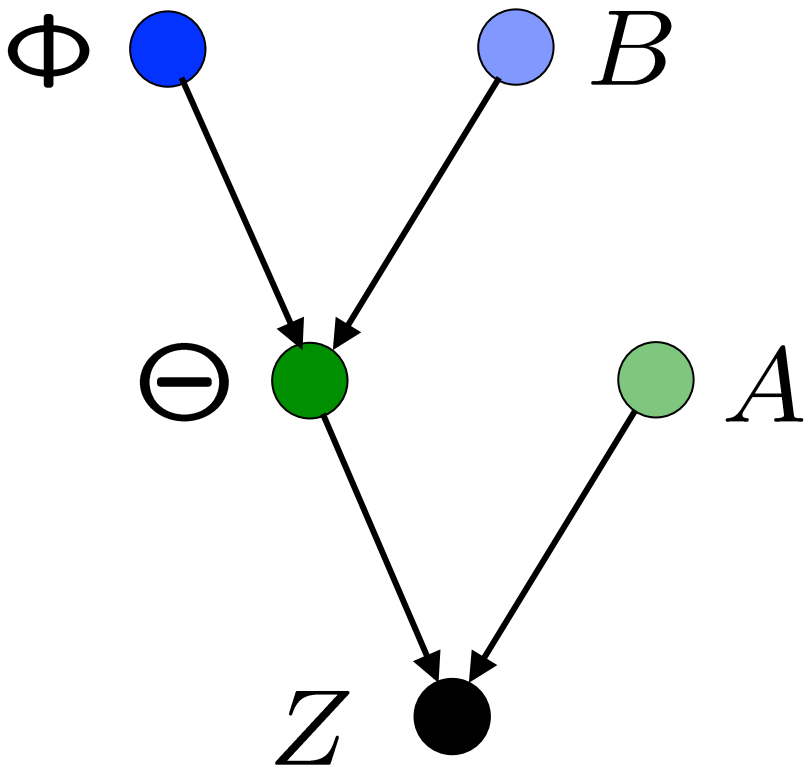
Graphical Model for Standard EM



Z	measurements
$\ominus$	planes
A	correspondences (hidden)

$$\text{JPD} = p(Z|A,\ominus)p(\ominus)p(A)$$

Graphical Model for Hier. EM



Z	measurements
Θ	planes
A	correspondences (hidden)
B	correspondences (hidden)
Φ	main directions

$$\text{JPD} = p(Z|A,\Theta)p(\Theta|B,\Phi)p(\Phi)p(A)p(B)$$

Expectation Maximization

- Goal: Find Θ and Φ so that JPD is maximized
- Take expectation over hidden variables
- Start with initial planes and main directions
- Iterate until convergence:

$$(\Theta^{[i+1]}, \Phi^{[i+1]}) =$$
$$\text{argmax}_{(\Theta,\Phi)} E_{AB}\left[\log p(A, B, \Phi, \Theta, Z) \mid \Theta^{[i]}, \Phi^{[i]}\right]$$

The E-Step in Hierarchical EM

$$p(A, B, \Phi, \Theta, Z) \propto \exp\left\{ -\frac{1}{2} \sum_n \sum_m \alpha_{nm} \left(\frac{d_1(\mathbf{z}_n, \theta_m)}{\rho} \right)^2 - \frac{1}{2} \sum_m \sum_k \beta_{mk} \left(\frac{d_2(\theta_m, \varphi_k)}{\sigma} \right)^2 \right\}$$

$$E[\alpha_{nm} \mid \Theta] = \frac{\exp\left\{ -\frac{1}{2} \left(\frac{d_1(\mathbf{z}_n, \theta_m)}{\rho} \right)^2 \right\}}{\sum_j \exp\left\{ -\frac{1}{2} \left(\frac{d_1(\mathbf{z}_n, \theta_j)}{\rho} \right)^2 \right\}}$$

$$E[\beta_{mk} \mid \Phi] = \frac{\exp\left\{ -\frac{1}{2} \left(\frac{d_2(\theta_m, \varphi_k)}{\sigma} \right)^2 \right\}}{\sum_j \exp\left\{ -\frac{1}{2} \left(\frac{d_2(\theta_j, \varphi_k)}{\sigma} \right)^2 \right\}}$$

The E-Step in Hierarchical EM

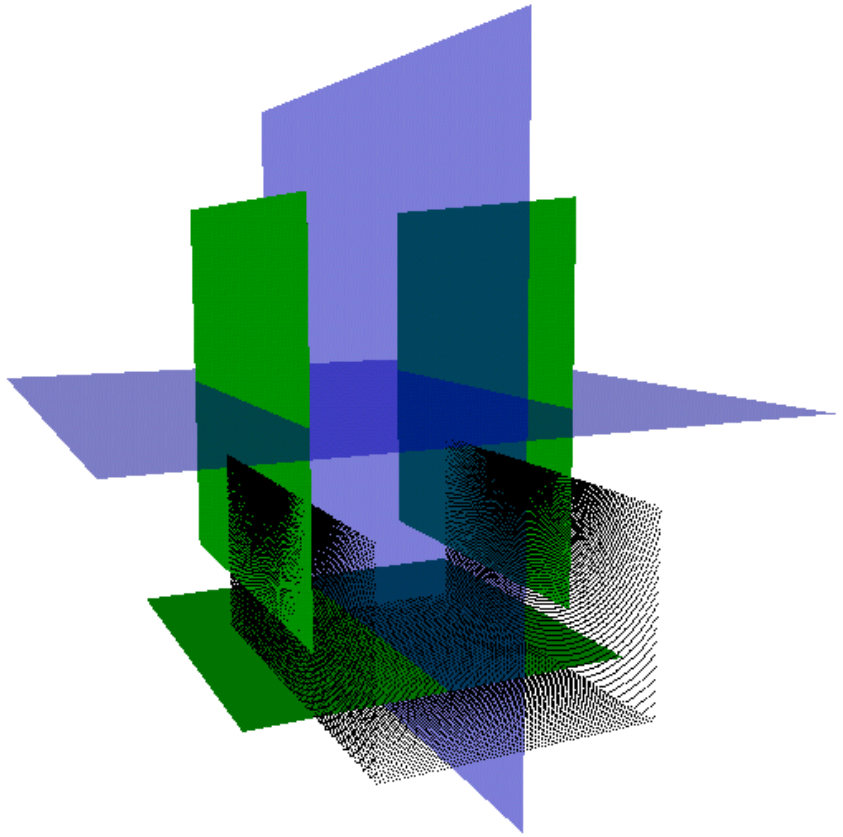
$$p(A, B, \Phi, \Theta, Z) \propto \exp\left\{ -\frac{1}{2} \sum_n \sum_m \alpha_{nm} \left(\frac{d_1(\mathbf{z}_n, \theta_m)}{\rho} \right)^2 \right\}$$

$$E[\alpha_{nm} \mid \Theta] = \frac{\exp\left\{ -\frac{1}{2} \left(\frac{d_1(\mathbf{z}_n, \theta_m)}{\rho} \right)^2 \right\}}{\sum_j \exp\left\{ -\frac{1}{2} \left(\frac{d_1(\mathbf{z}_n, \theta_j)}{\rho} \right)^2 \right\}}$$

The M-Step

- Find planes and main directions so that the expected log-likelihood is maximized
- No closed-form solution (LL is non-linear)
- Optimization technique: conjugate gradient based method

Example



84,660 points

$\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{84,660}$

3 planes

$\theta_1, \theta_2, \theta_3$

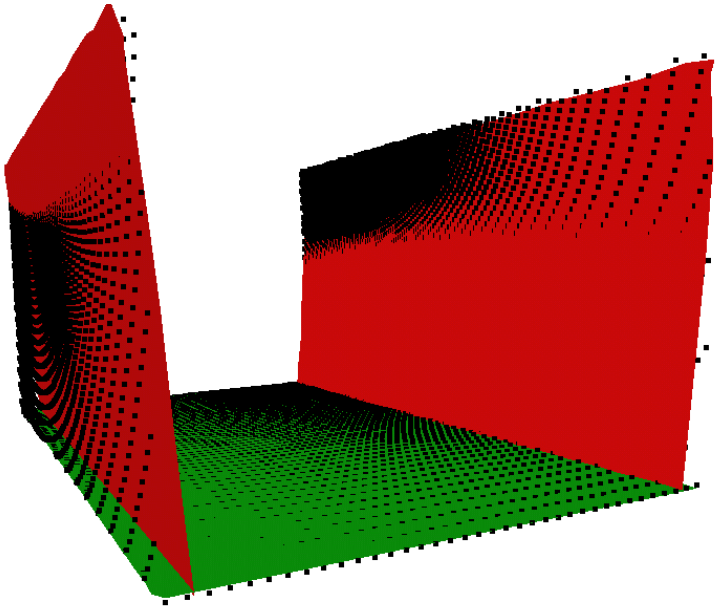
2 main directions

Φ_1, Φ_2

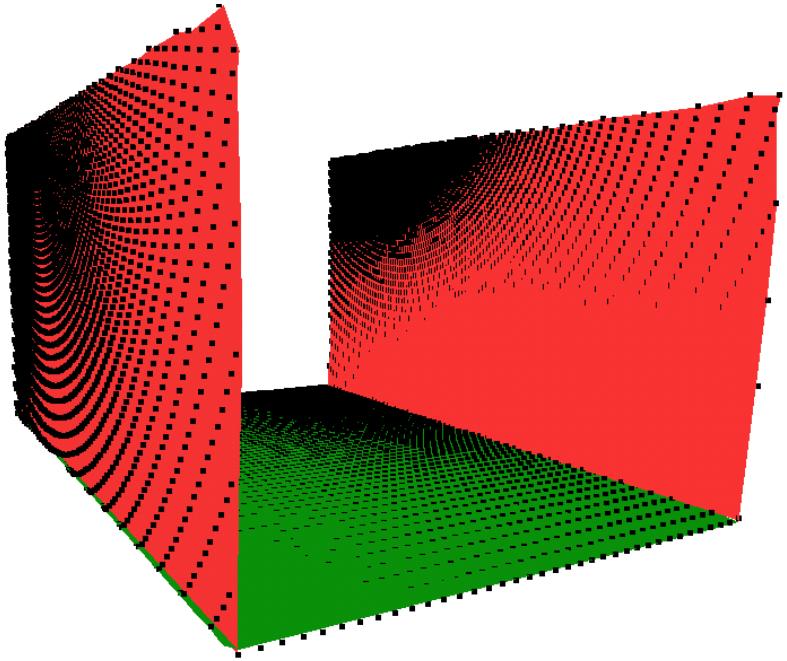
Main directions are normal vectors of planes.

Typical Result

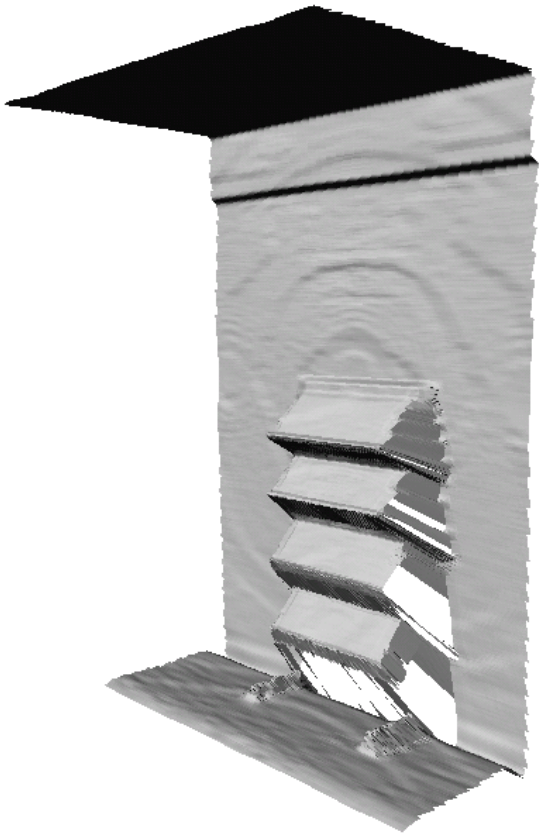
Standard EM:

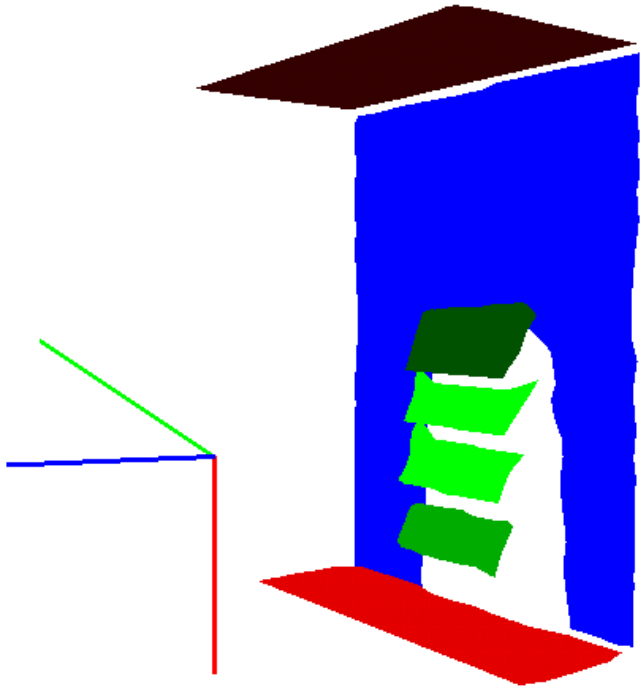


Hierarchical EM:

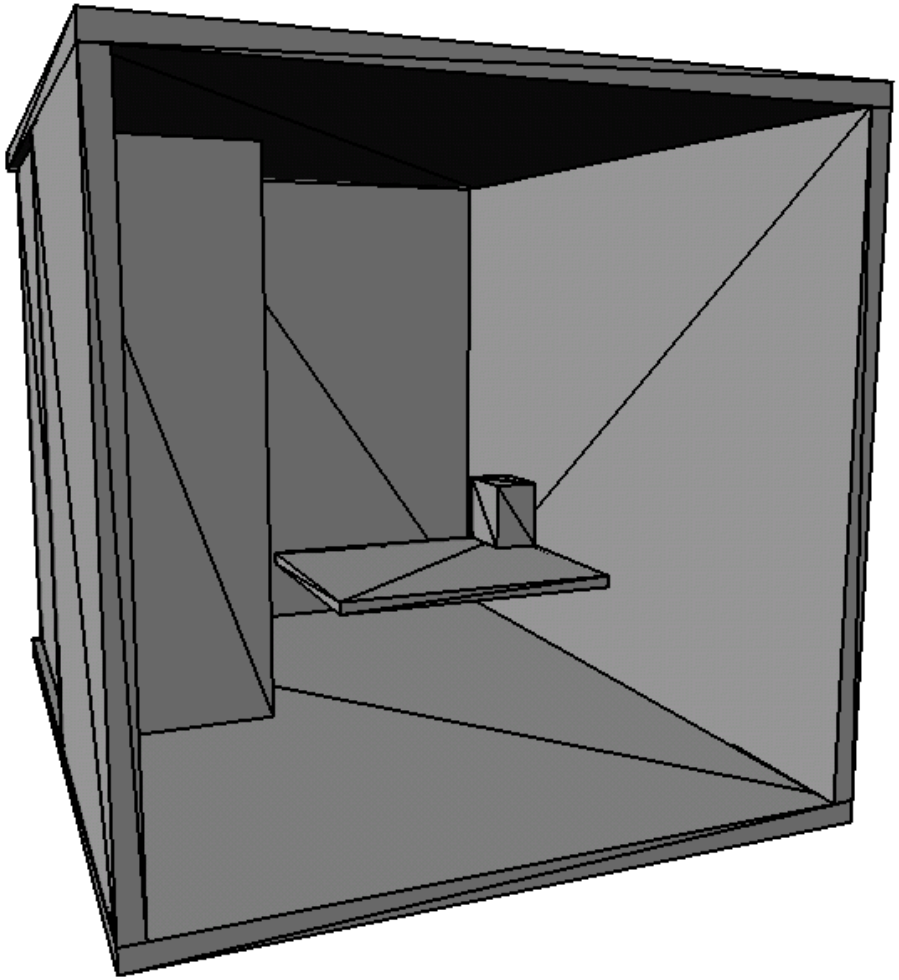


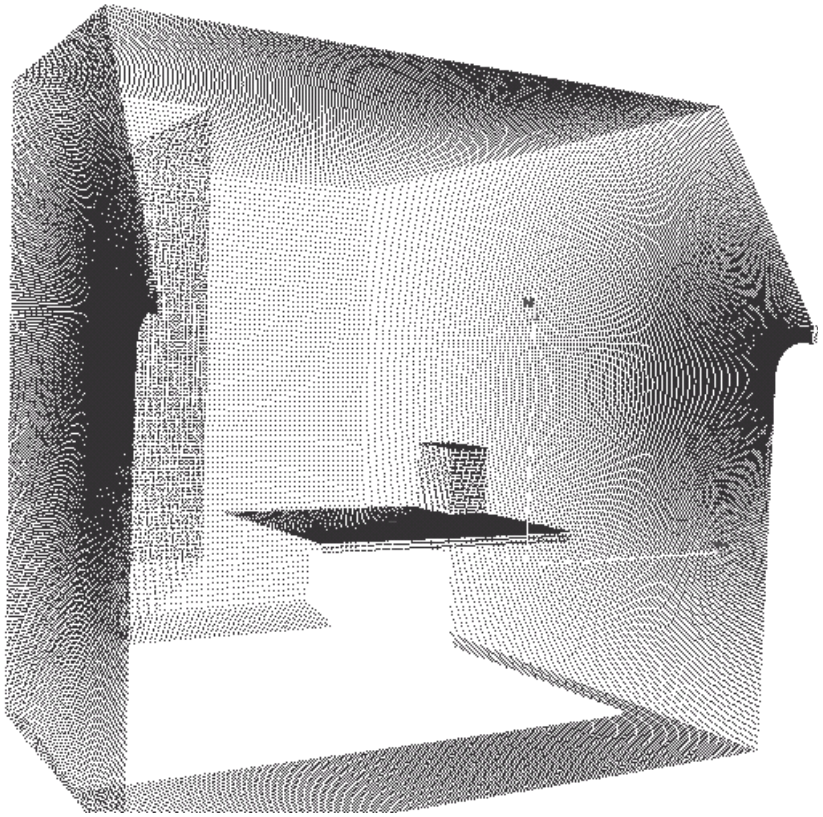
Real Data Experiment



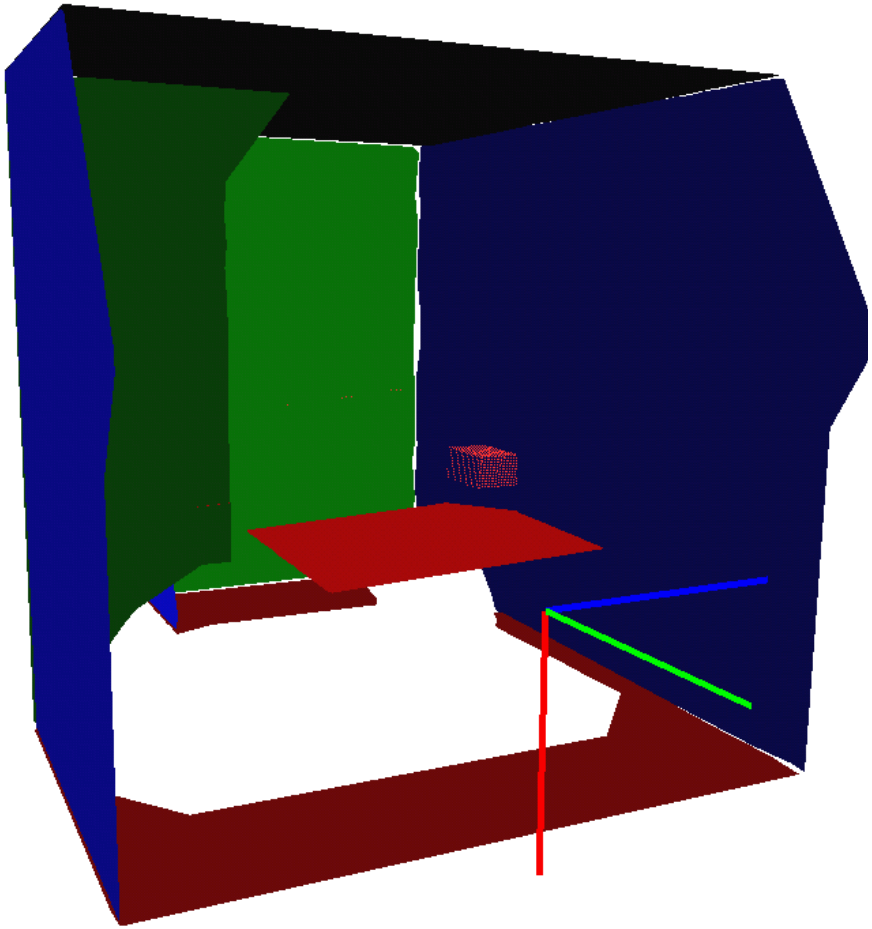


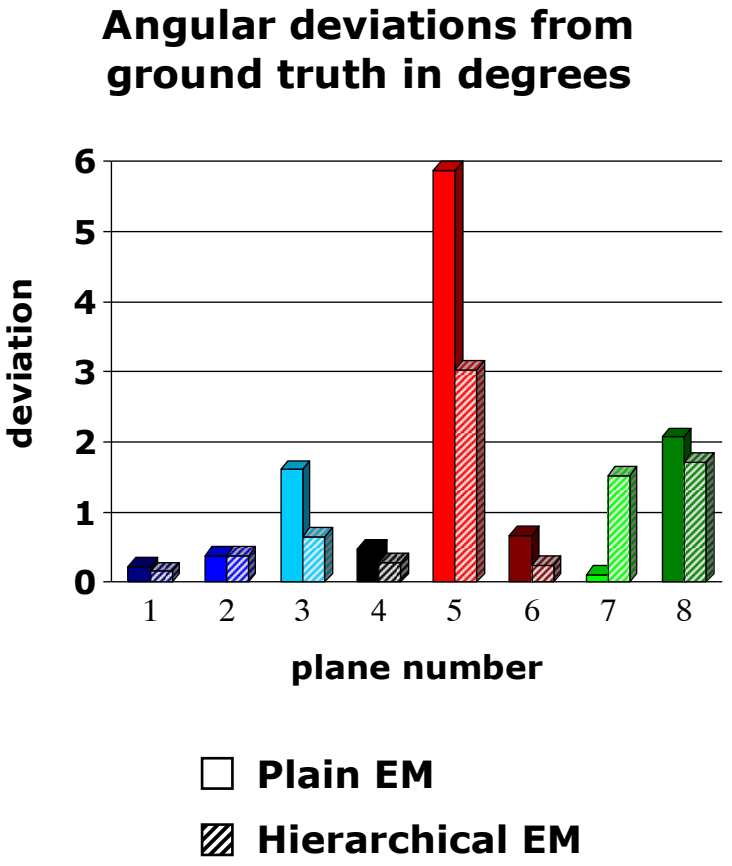
Quantitative Evaluation (1)





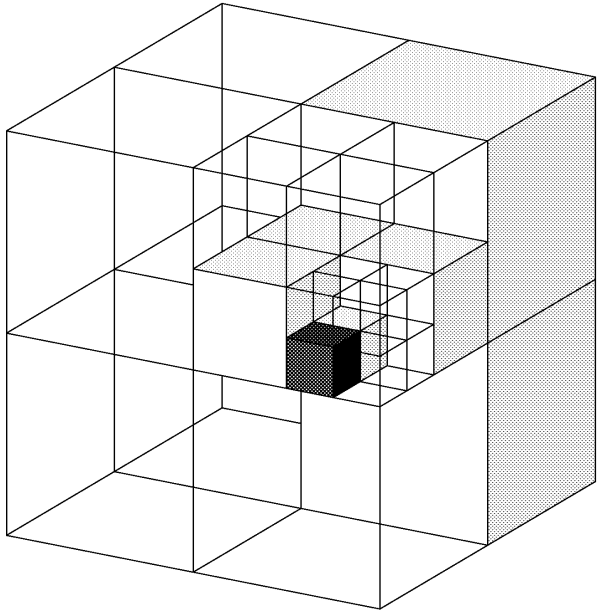
Quantitative Evaluation (2)

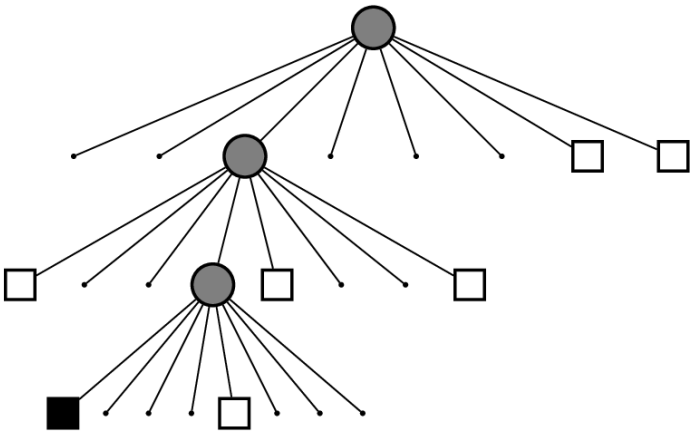




Map Representation by Octrees

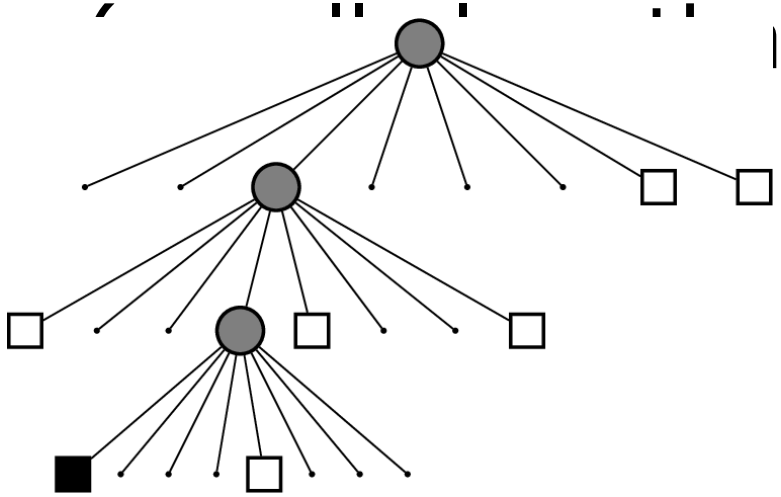
- Tree-based data structure
- Recursive subdivision of space into octants
- Volumes allocated as needed
- Multi-resolution

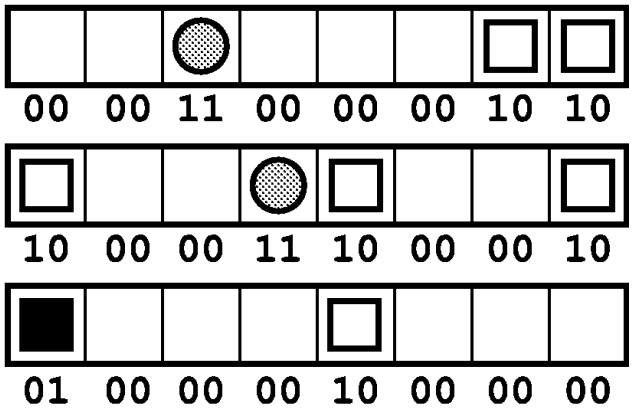




Map Files

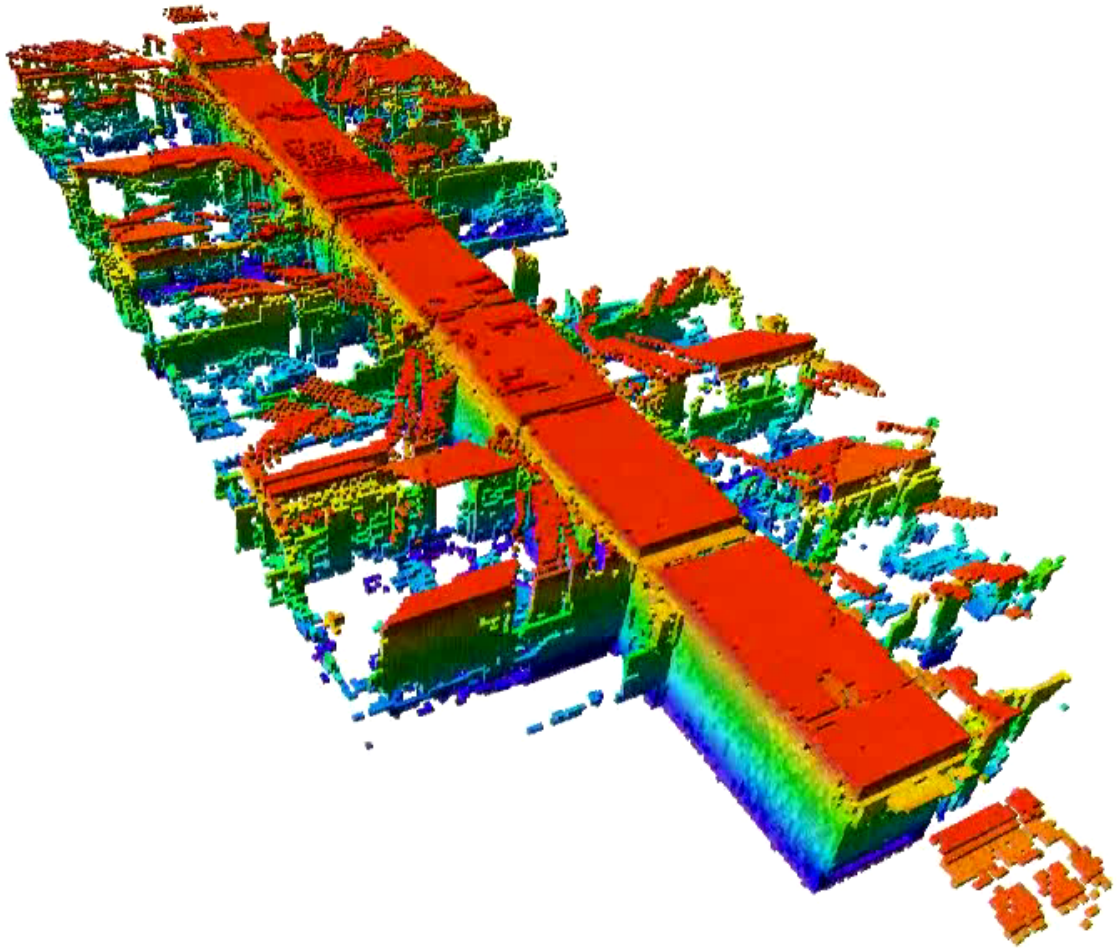
- Maximum-likelihood map stored as compact bitstream
- Occupied, free, and unknown areas
- Very moderate space requirements (n 2 MB)





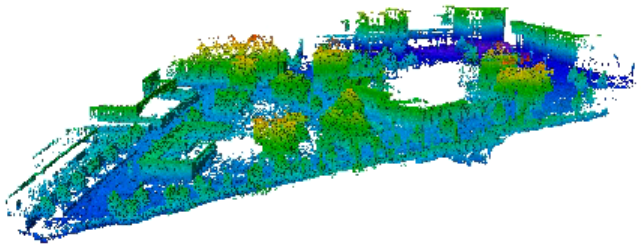
Examples: Office Building

- Freiburg, building 079



Examples: Large Outdoor Areas

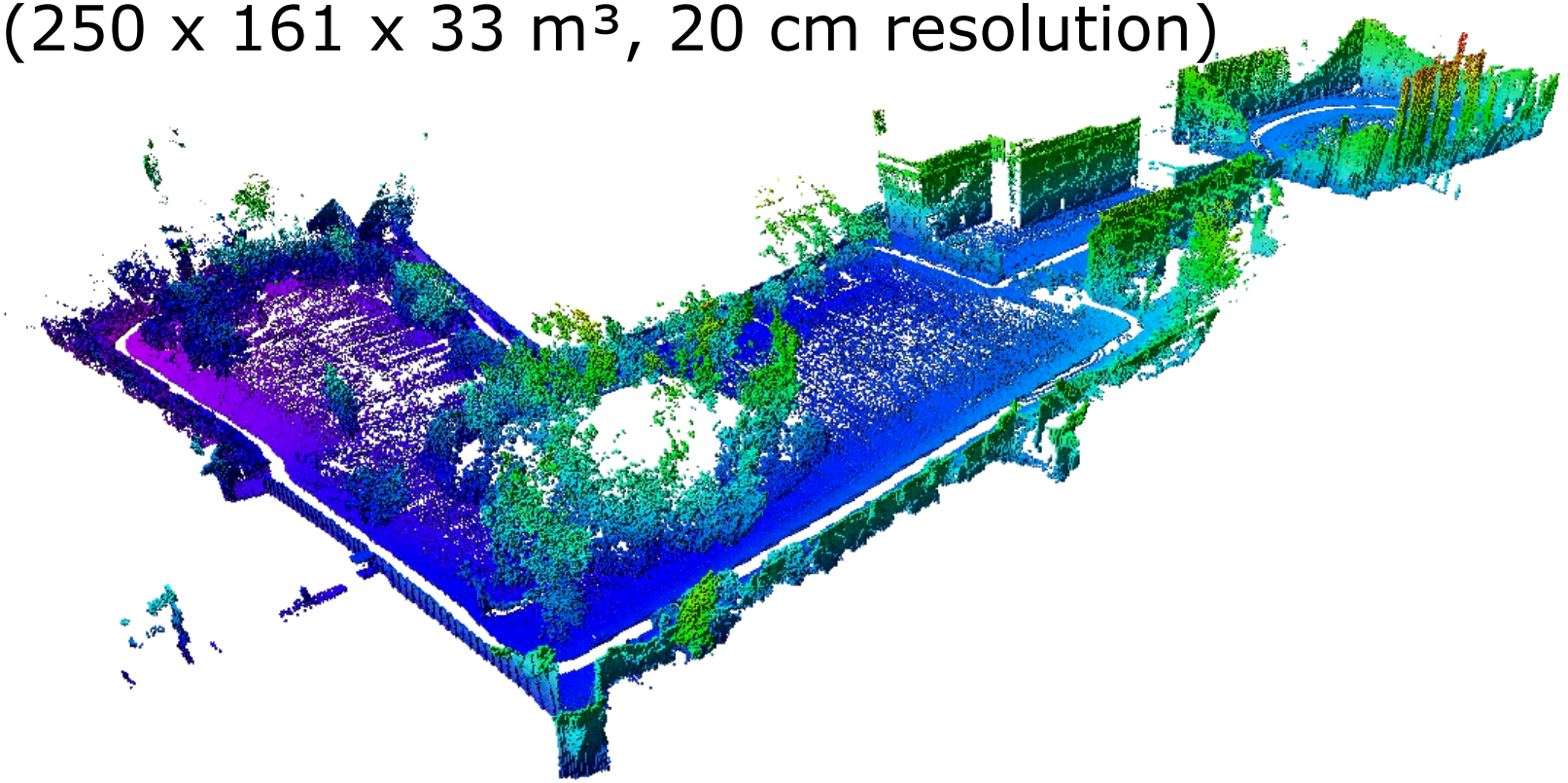
- Freiburg computer science campus
  (292 x 167 x 28 m³, 20 cm resolution)



Examples: Large Outdoor Areas

- Oxford *New College* dataset (Epoch C)

  (250 x 161 x 33 m³, 20 cm resolution)



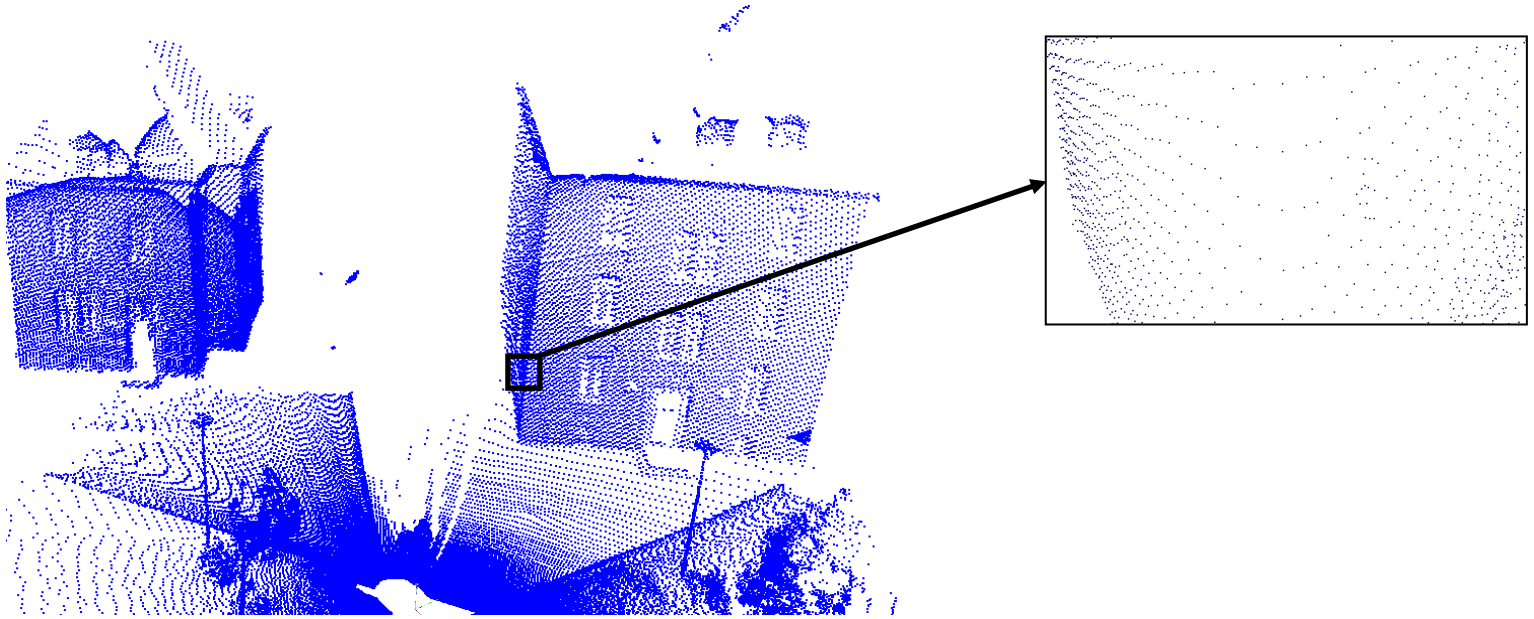
Memory Usage

Map dataset	Mapped area [m^3]	Resolution [m]	Memory consumption [MB]			File size [MB]	
			Full grid	No compr.	Lossless compr.	All data	Binary
FR-079 corridor	$43.8 \times 18.2 \times 3.3$	0.05	80.54	73.64	41.70	15.80	0.67
		0.1	10.42	10.90	7.25	2.71	0.14
Freiburg outdoor	$292 \times 167 \times 28$	0.20	654.42	188.09	130.39	49.75	2.00
		0.80	10.96	4.56	4.13	1.53	0.08
New College (Epoch C)	$250 \times 161 \times 33$	0.20	637.48	91.43	50.70	18.71	0.99
		0.80	10.21	2.35	1.81	0.64	0.05

http://octomap.sf.net

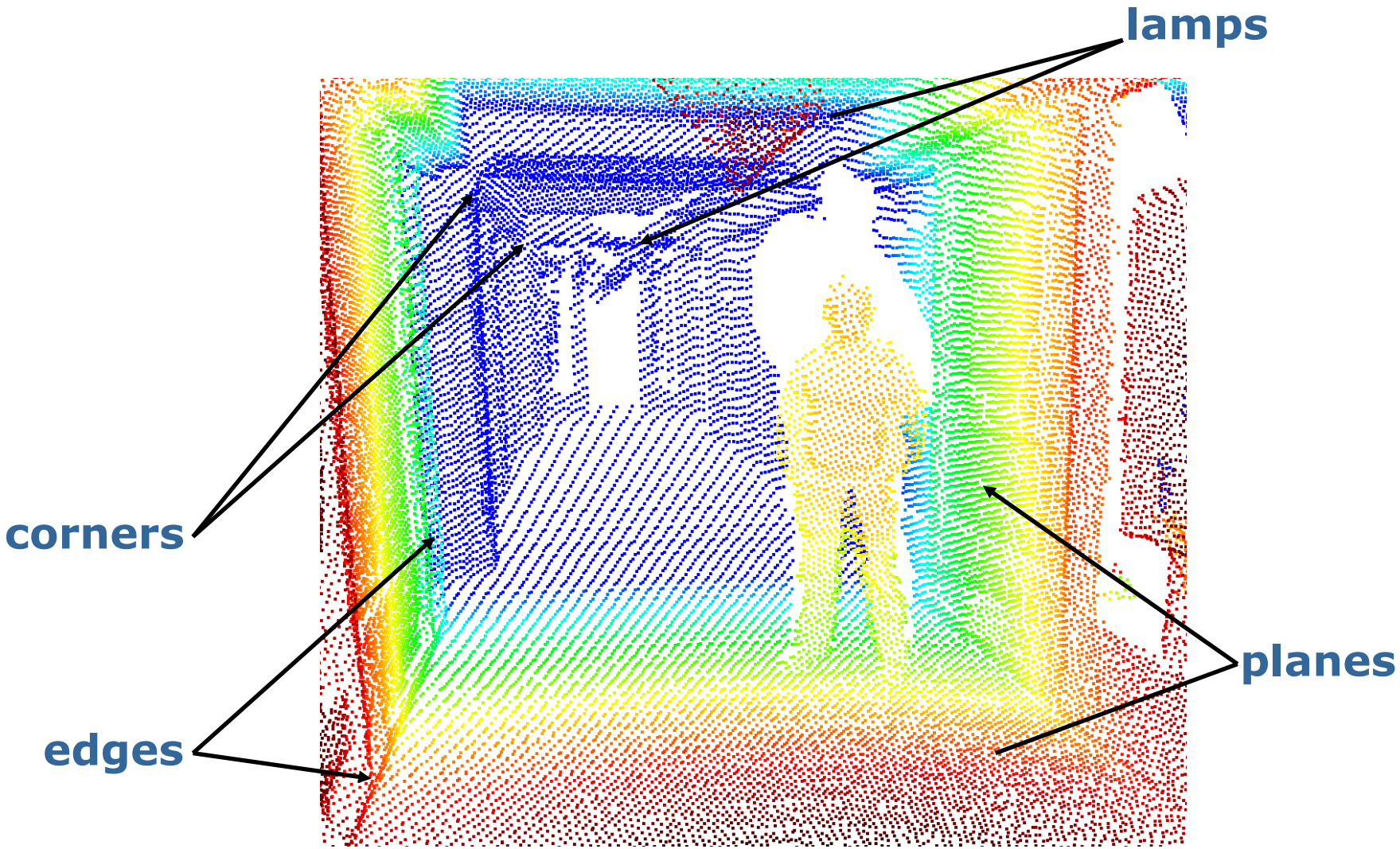
Unsupervised Compression

- 3D scanners provide point clouds sampled from the surfaces in the environment



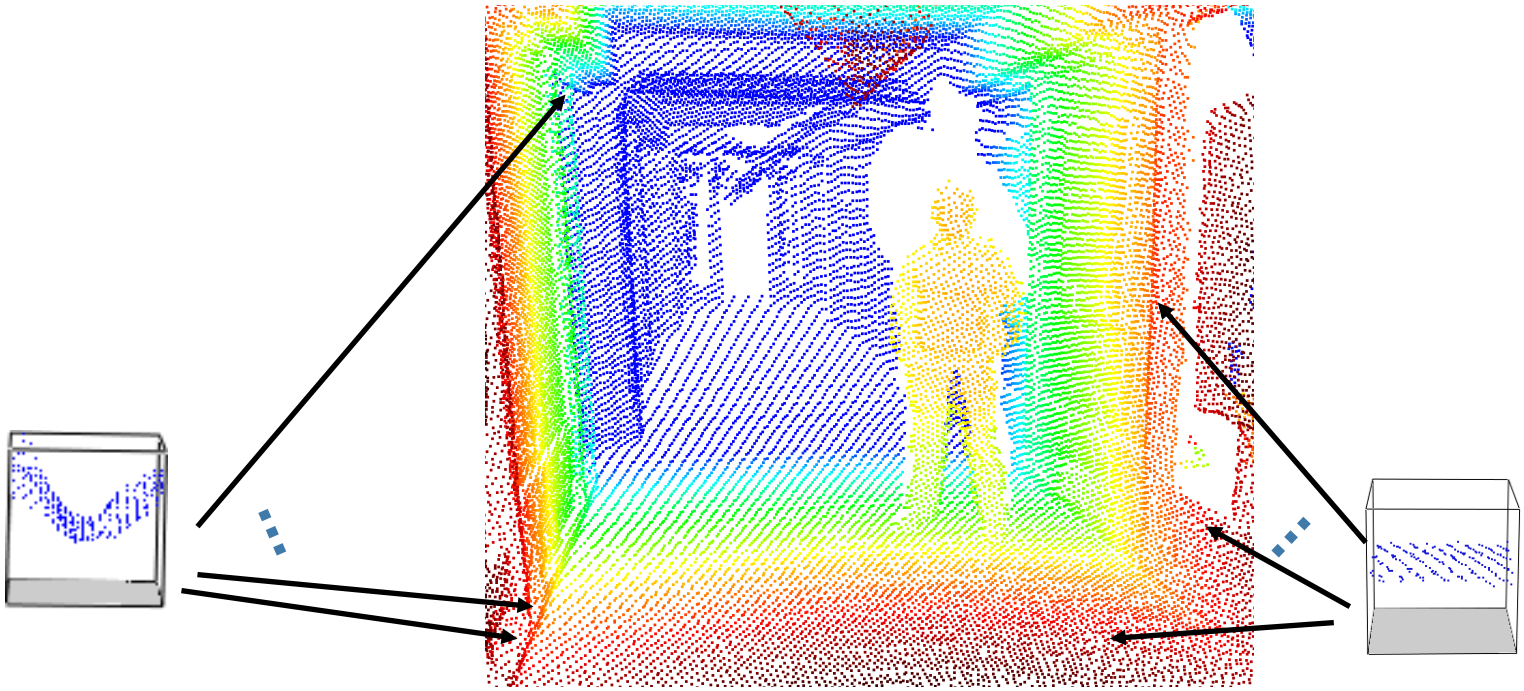
- Point clouds are low-level representations.
- Goal: Approach that enables us to reason about surface structure even without given models.

Recurrent Structure



Key Idea

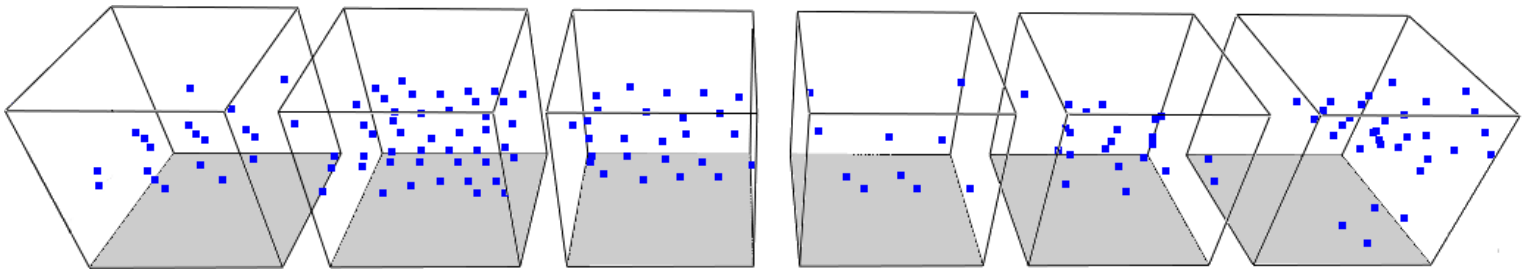
- Model the environment by a set of primitives representing similar structures
  - simplified and more efficient reasoning
  - lower storage requirements



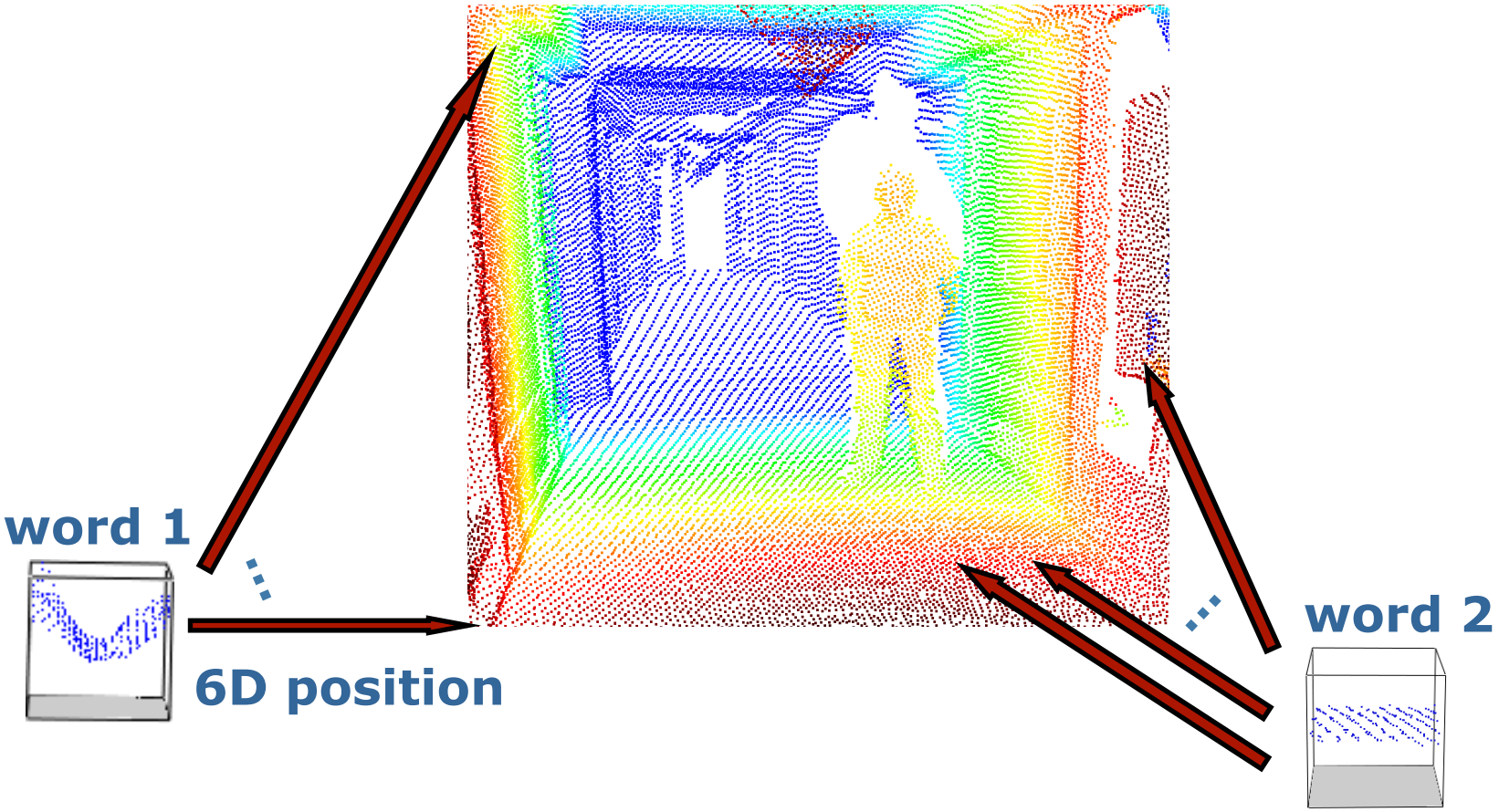
Model

- Model $M(P, A)$
  - $A = \{a_i\}$: Set of words
  - P : Set of 6D word positions

  - Word a_i
    - Surface sample of fixed size
    - The surface is described by small point clouds

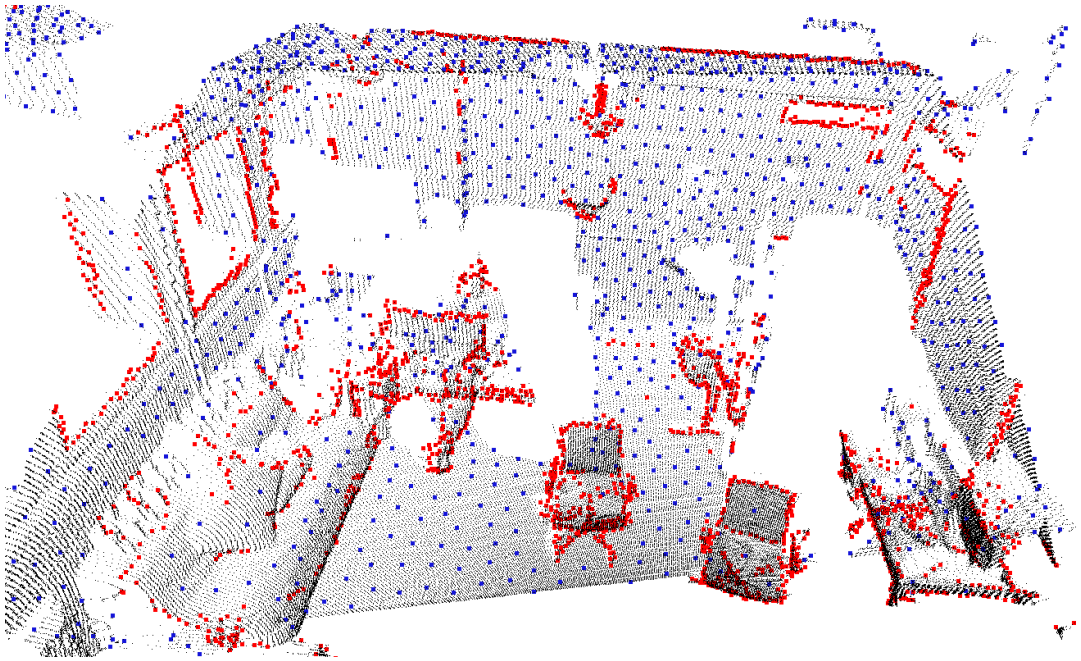


Model



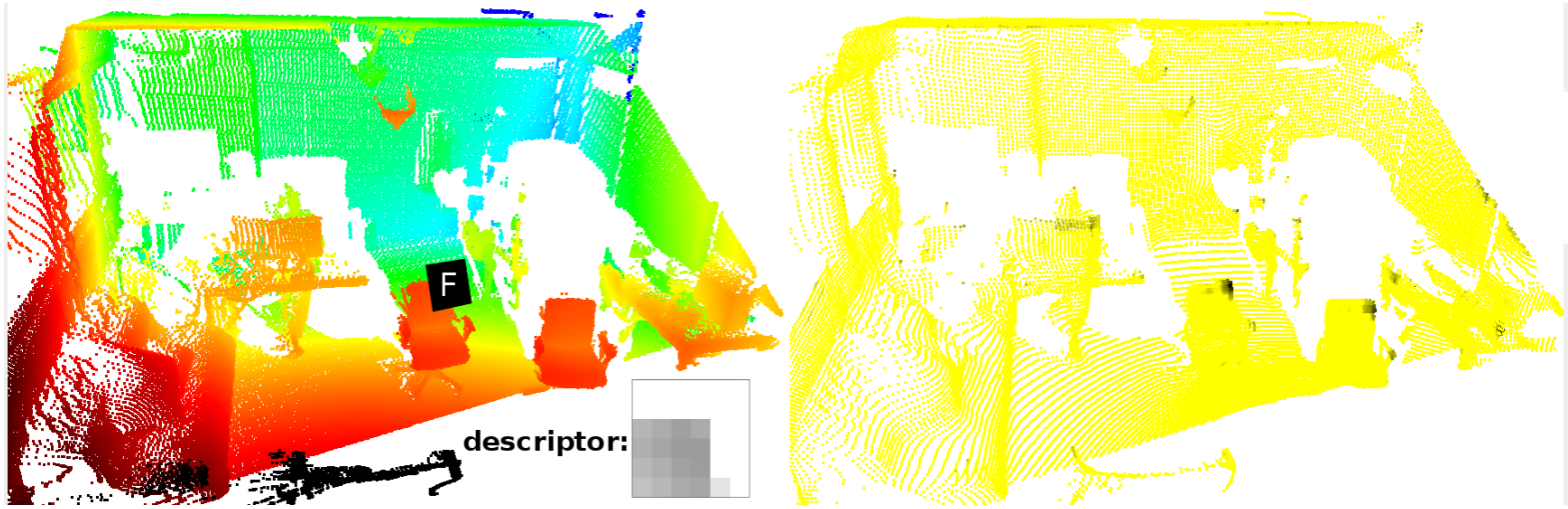
How to Get Relevant Words?

- We extract words from the individual scans
- and sample the corresponding locations according to
  - interest points (structure)
  - uniform sampling (no structure).



Computing Words at Interest Points

- Extract invariant local 6DoF coordinate frames
  - 3D position as origin of the word
  - Orientation according to normal and aligned to z-axis
- Compute a range image feature descriptor for fast similarity tests [Steder et al., ICRA 09]
- Store local point cloud around the interest point



Application: Compressing Scenes

Idea: use a small set of words to represent the entire scene.

Questions:

- Which words to choose?
- How many words to choose?

We have to trade off accuracy vs. complexity.

Alphabet Selection

Model accuracy vs. model complexity

- Iteratively refine the alphabet
- Minimizing the **B**ayesian **I**nformation **C**riterion

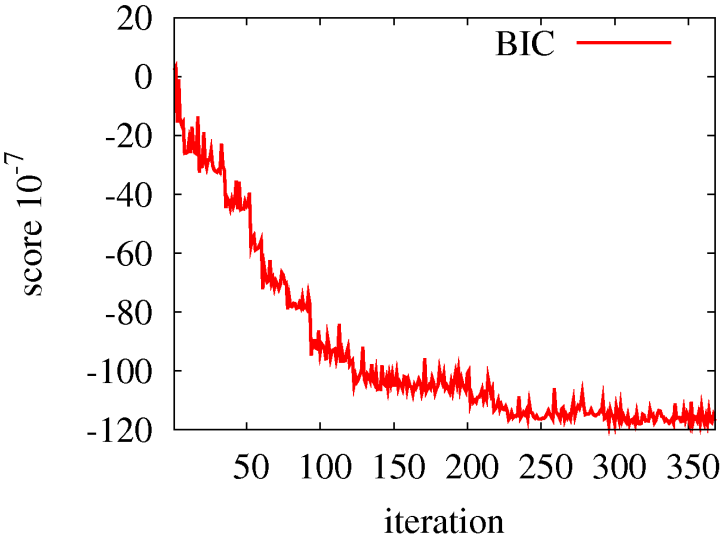
$$BIC(M,D) = -2 \cdot ln(L(M,D)) + n \cdot ln(N)$$

accuracy

complexity

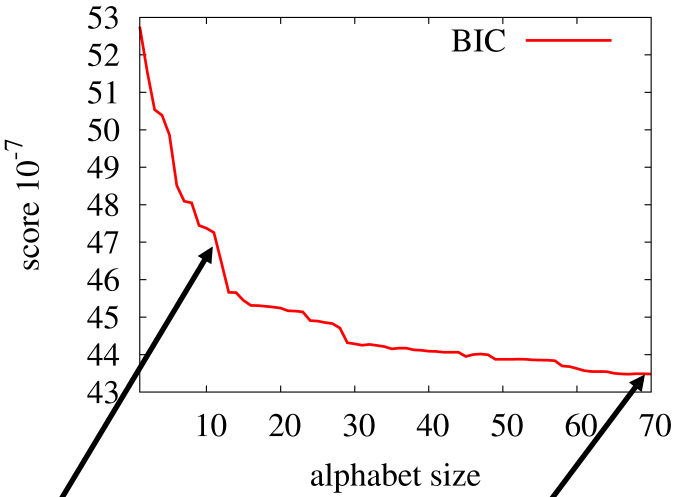
Resulting Algorithm

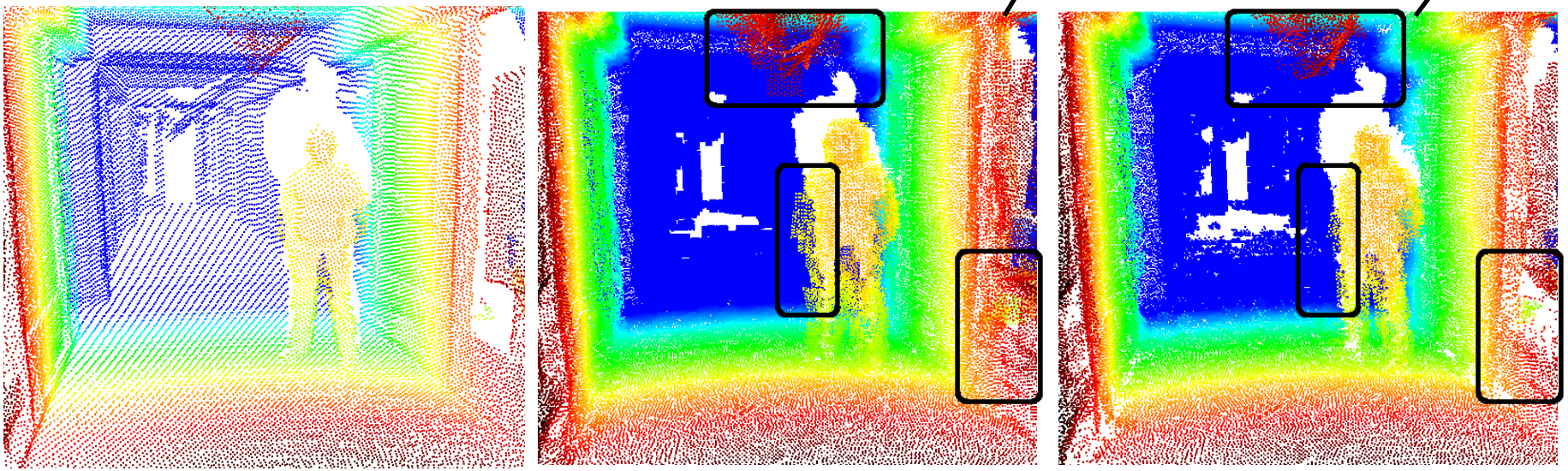
Start with an empty *A*
Iterate
    Add random word to *A*
    Construct approximation
    Calculate **BIC**
    If **BIC** gets worse
        Drop word
Every 10th iteration
    Delete random word from *A*
    If **BIC** gets worse
        Add word again



Compressing a Scene

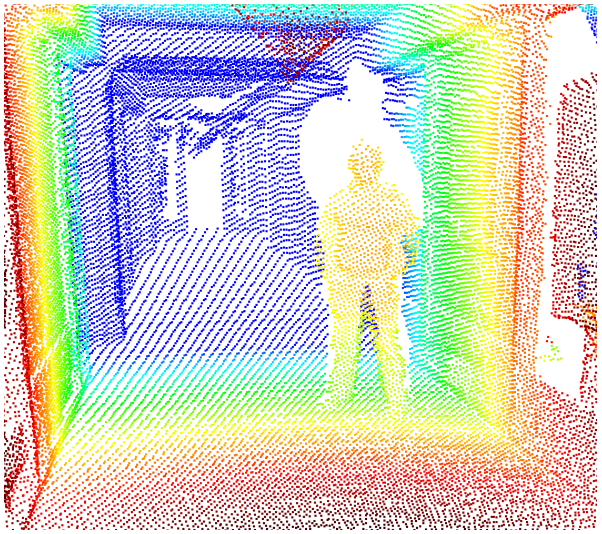
- Original number of words: 6,337
- Different words in model: 70
- Coverage: 95%
- Outliers: 7%
- Compression rate: 35.5%



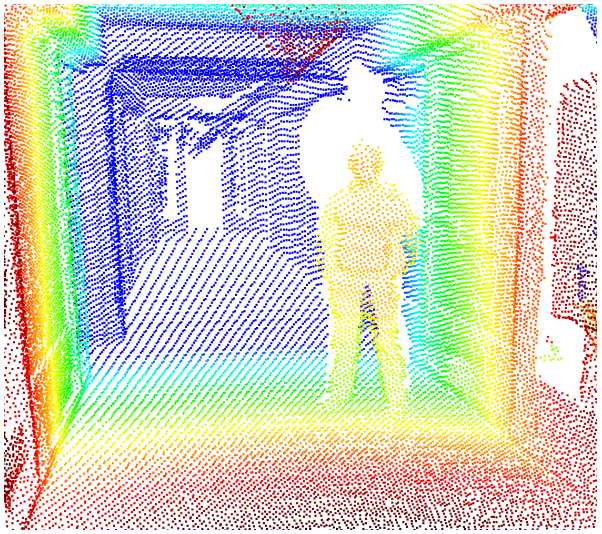


Original point cloud **Alphabet size 10** **Alphabet size 70**

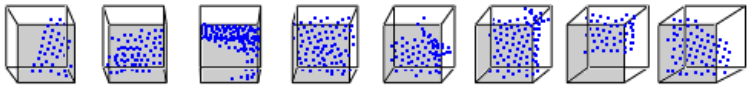
Reusing an Existing Alphabet to Represent a New Scene



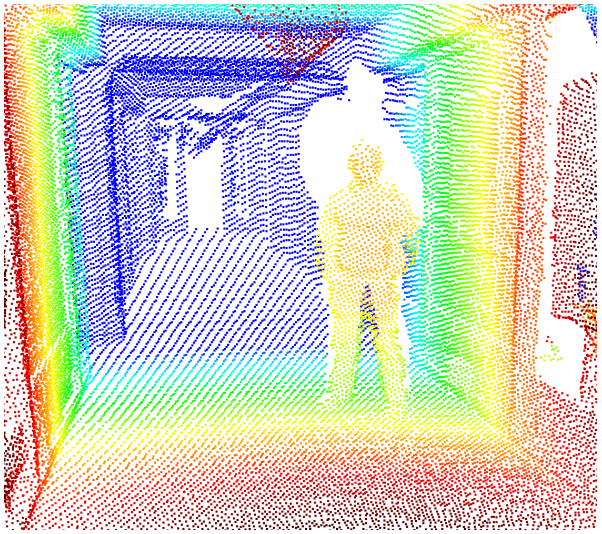
Reusing an Existing Alphabet to Represent a New Scene



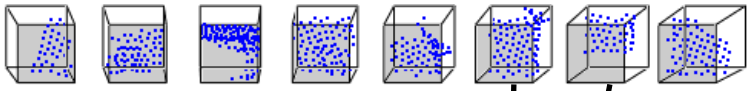
Extract the alphabet



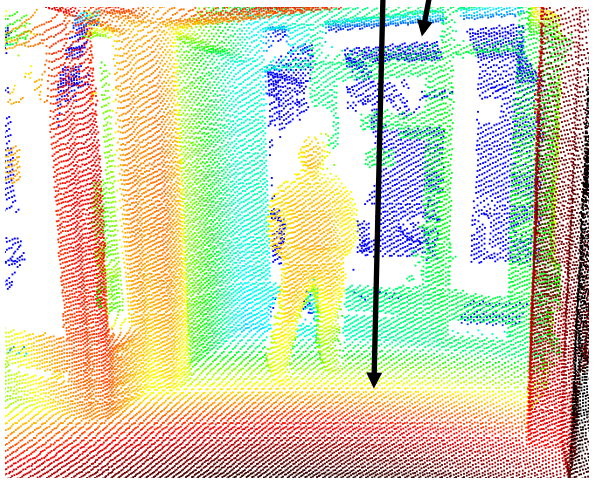
Reusing an Existing Alphabet to Represent a New Scene



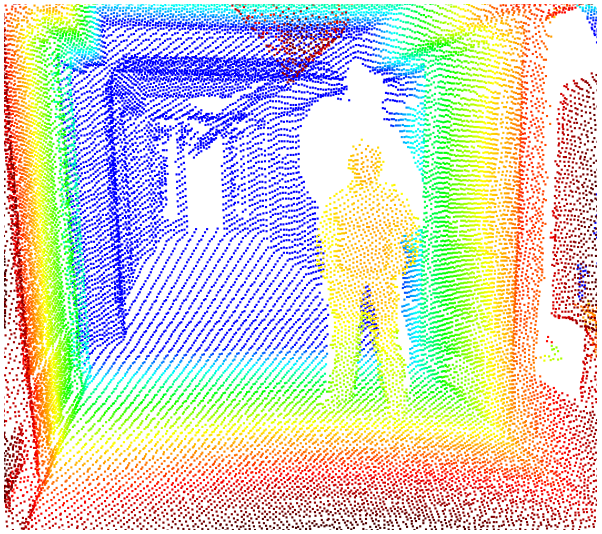
Extract the alphabet



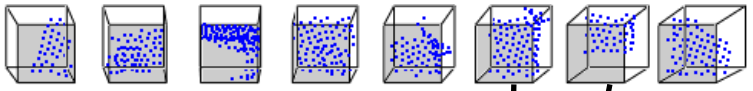
Apply alphabet to other scene



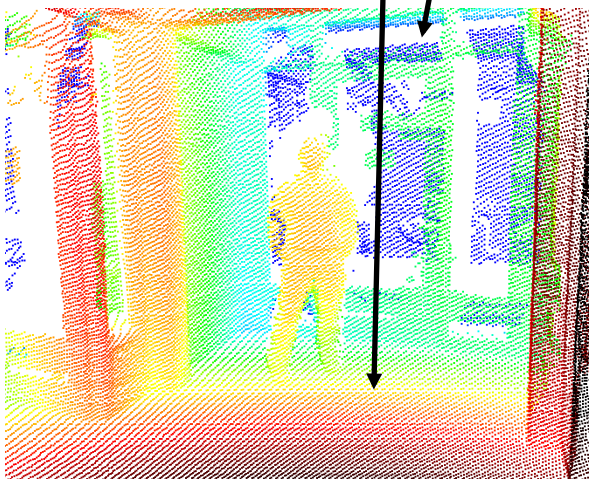
Reusing an Existing Alphabet to Represent a New Scene



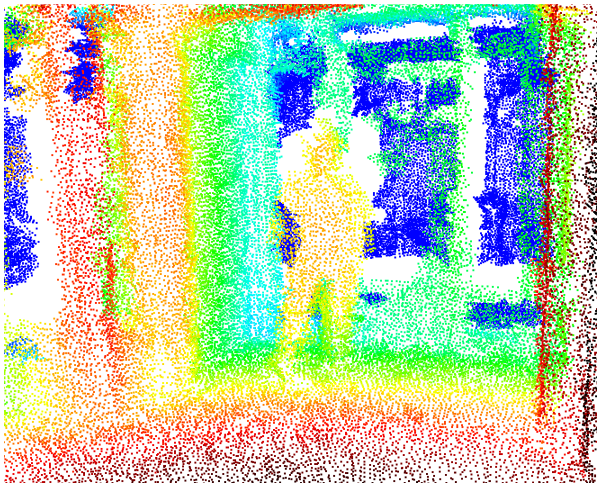
Extract the alphabet



Apply alphabet to other scene

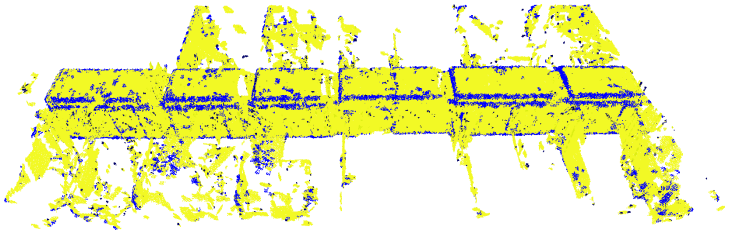


Reconstruct the scene

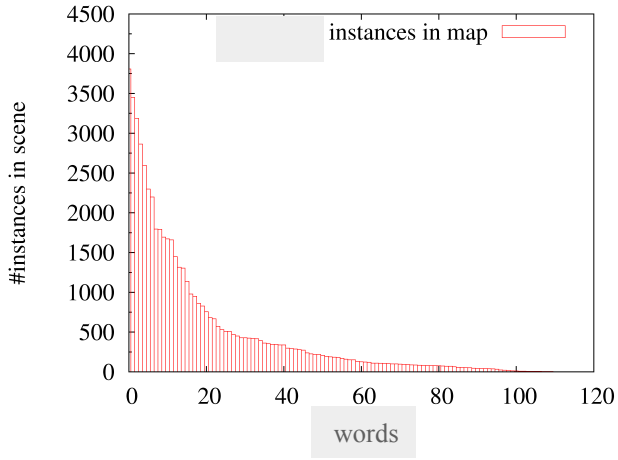


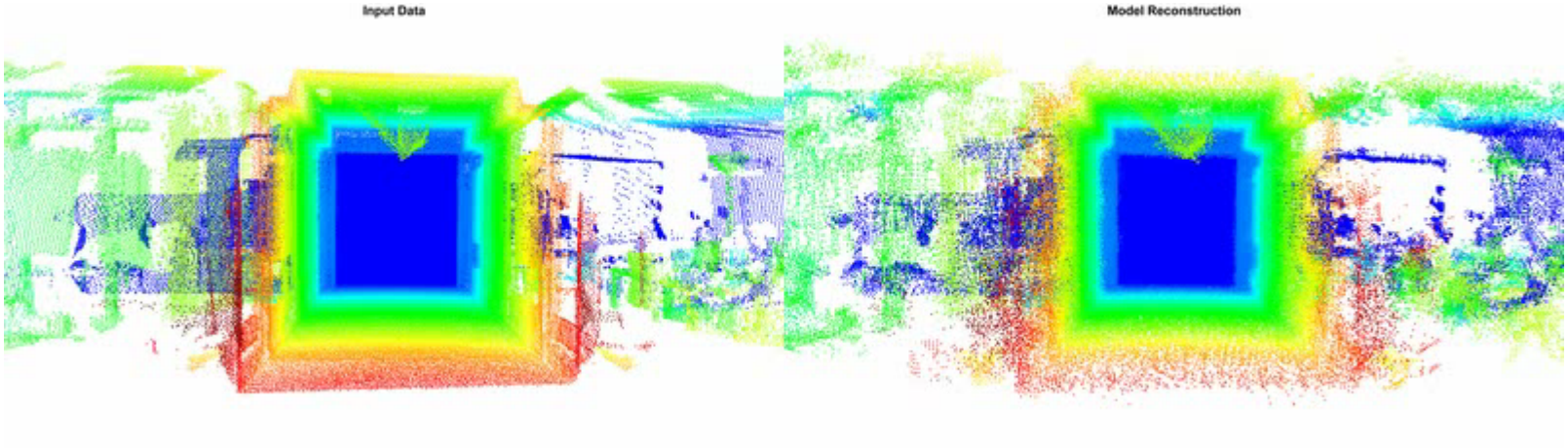
89% inliers and 15% outliers

Compressing Data of an Office Environment



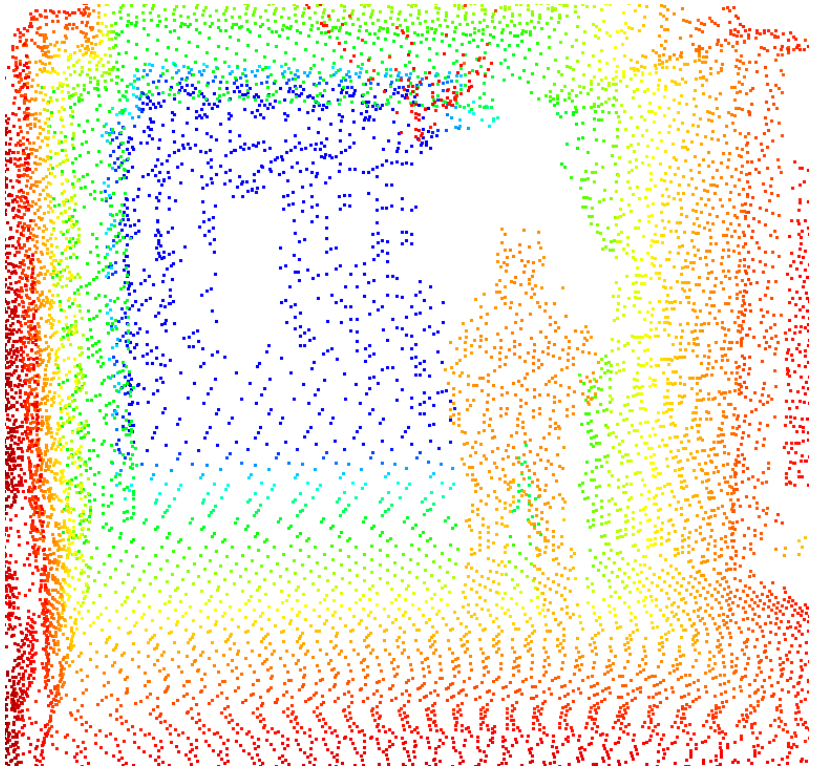
- Original number of words: 123,930
- Different words in model: 114
- Size: 20 → 3.5 MByte
- Compression rate: 16%



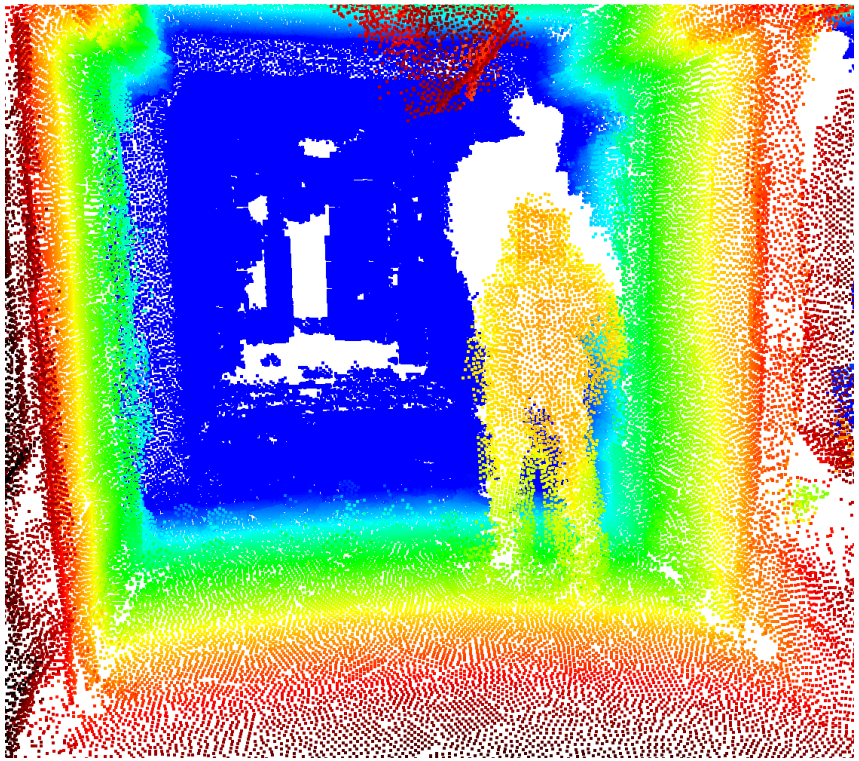


Comparison with Sub-sampling

Sub-sampled (33%)

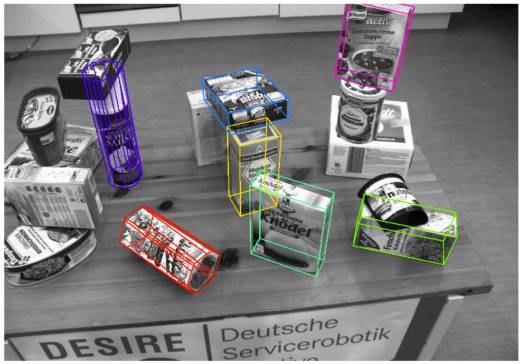


Our approach (35.5%)



Why Object Recognition?

- High-level reasoning
- Interaction
- Fetch and carry tasks
- Efficiency
- Compactness
- …



Vision vs. 3D

Vision:

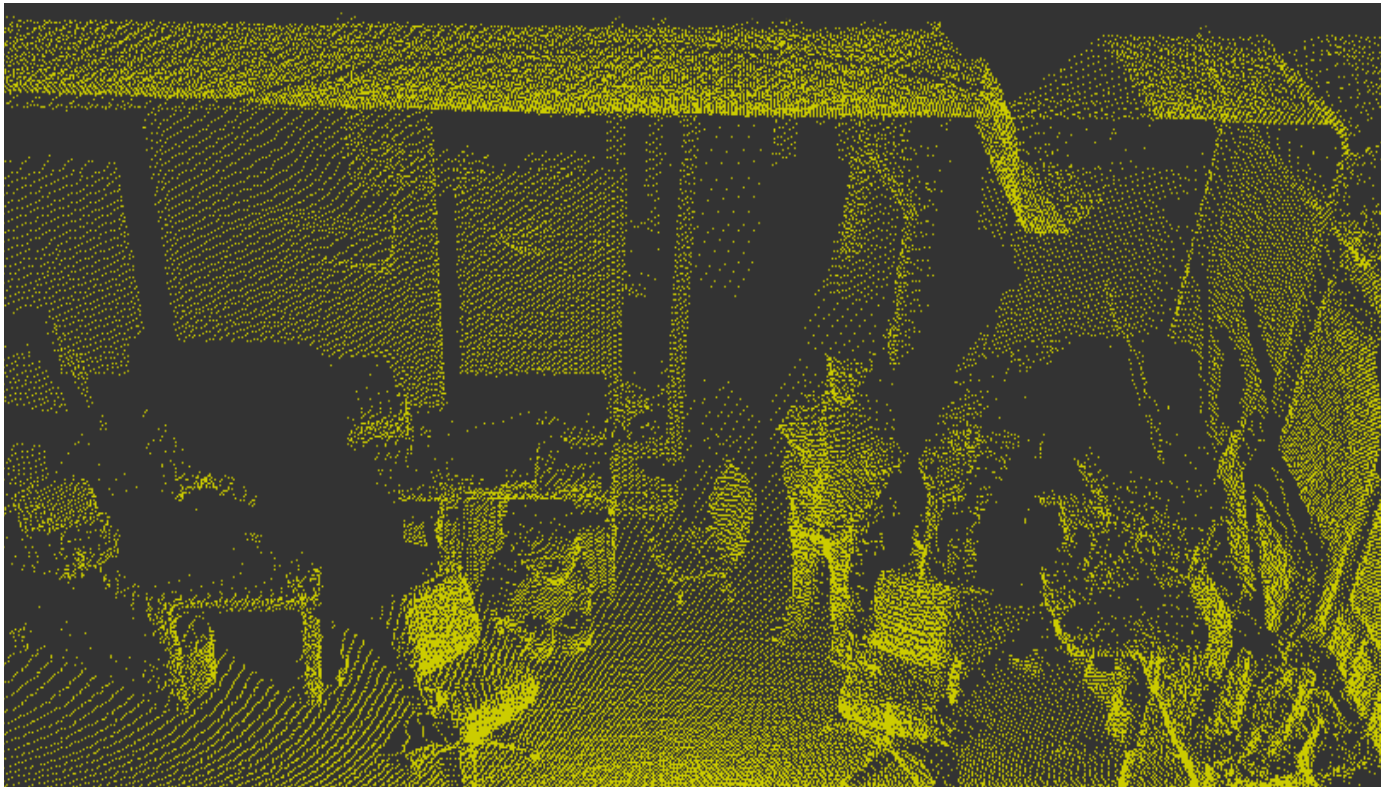
- Sift and variants
- Bag of words
- Impressive applications
  - structure from motion
  - image retrieval
  - scene interpretation

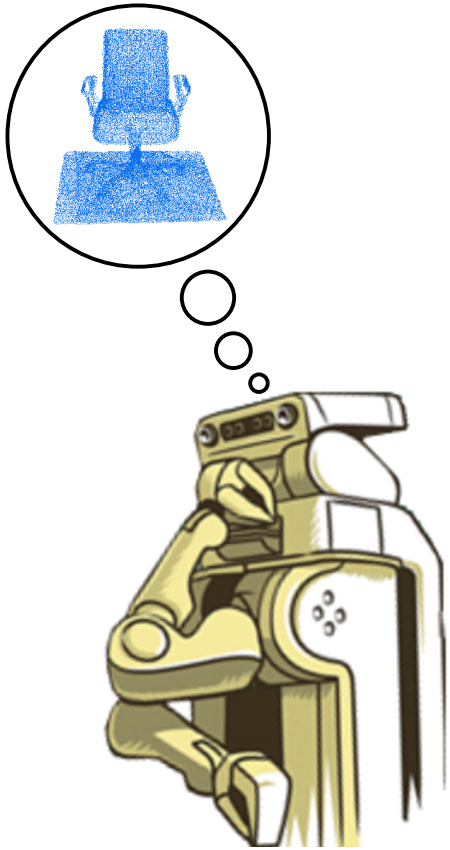
successful!

3D

- Spin images
- Shape distributions
- …

Object Recognition and Localization in 3D Range Data





Why is it Hard?

- high dimensionality
- huge amount of data
- complexity of similarity checks/score calculations
- non-trivial weighing of data points (resolution dependence)
- …

Two Steps

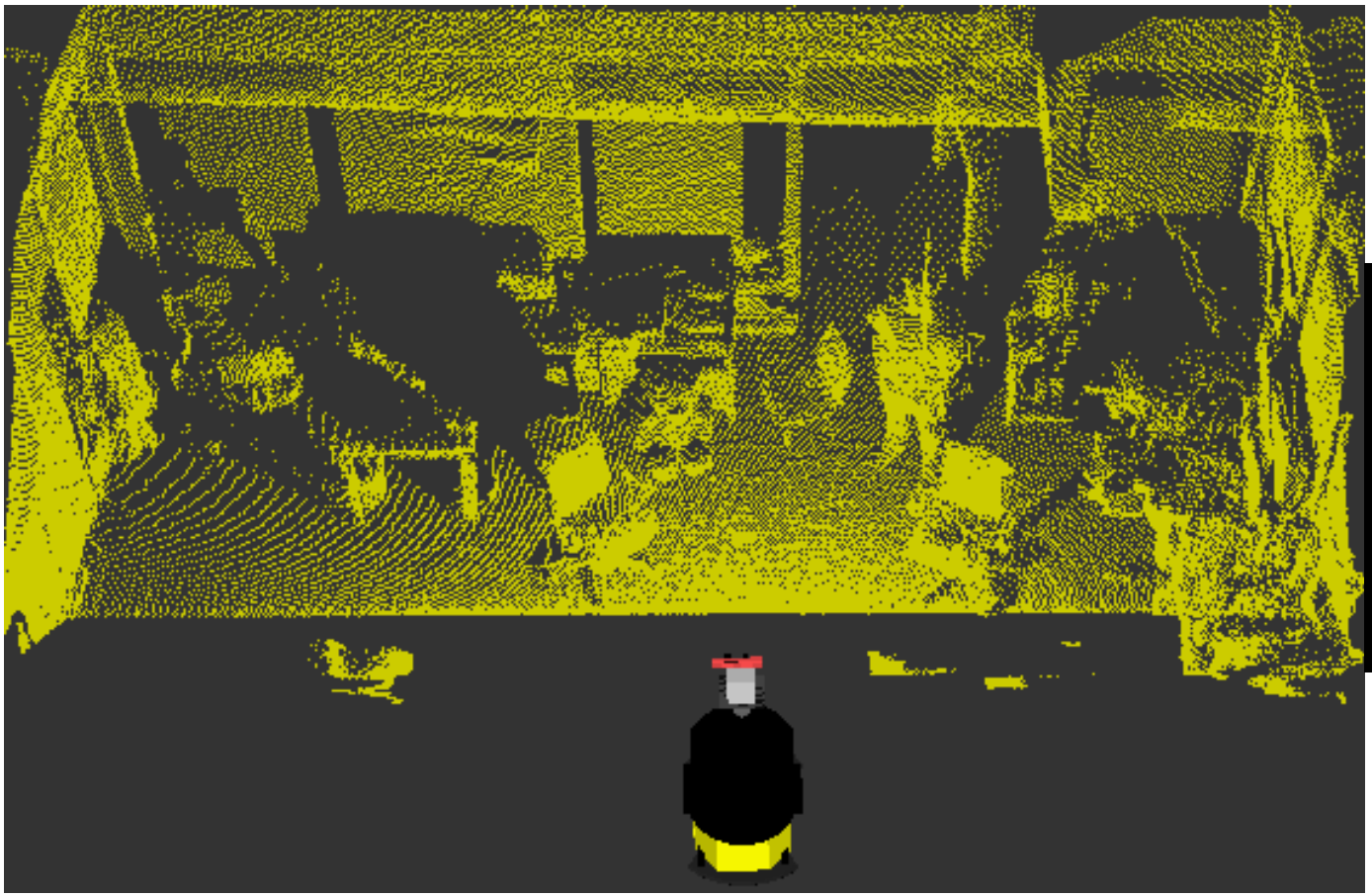
- **Detection of candidate poses:**
  Detect possible positions where the object might be.
  (fast and selective – no false negatives, few false positives)

- **Scoring:**
  Spatial verification step that tests how plausible the candidate poses are according to the sensor data.

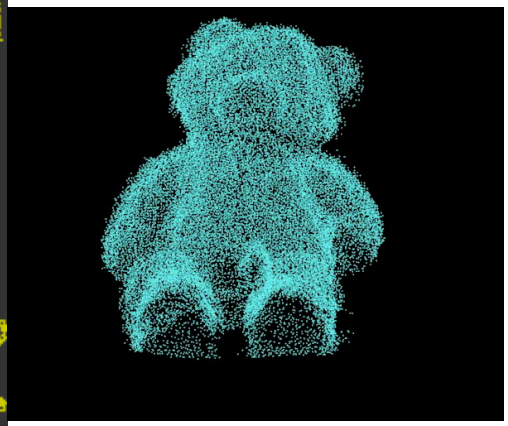
Model creation

- Models of the object can come from
  - Multiple merged scans of the object
  - CAD models
  - 3D databases (e.g., Google 3D warehouse)
  - …

Example

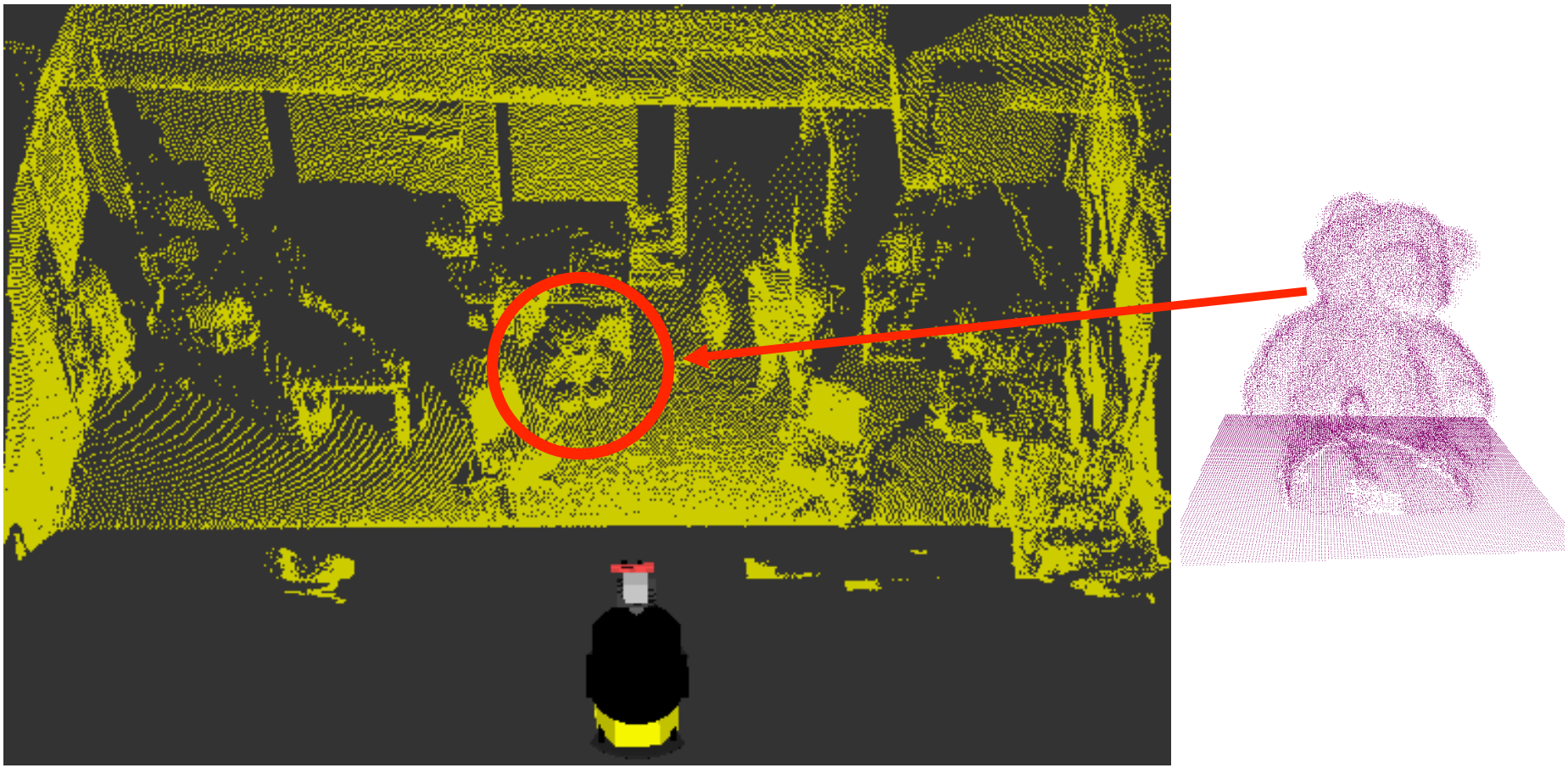


Scene



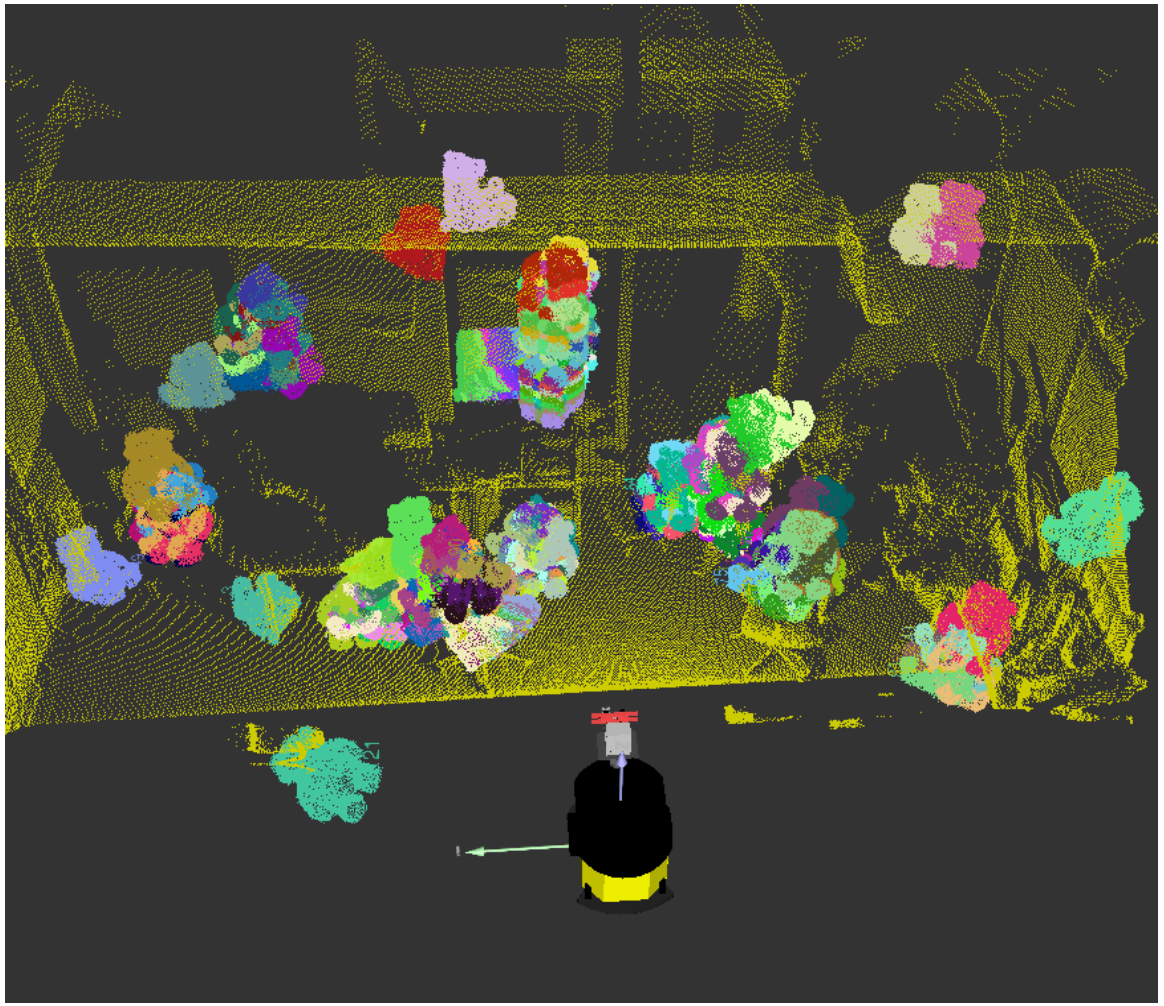
Model

Example



Candidate Poses

- Assuming we have a number of candidate positions



Candidate transformation

- A candidate transformation T describes a translation and rotation between the model and the scene.
- T can, e.g., be a 4x4 matrix.
- For a point m in the model, $s = T \cdot m$ is the corresponding point in the scene.

Score Calculation

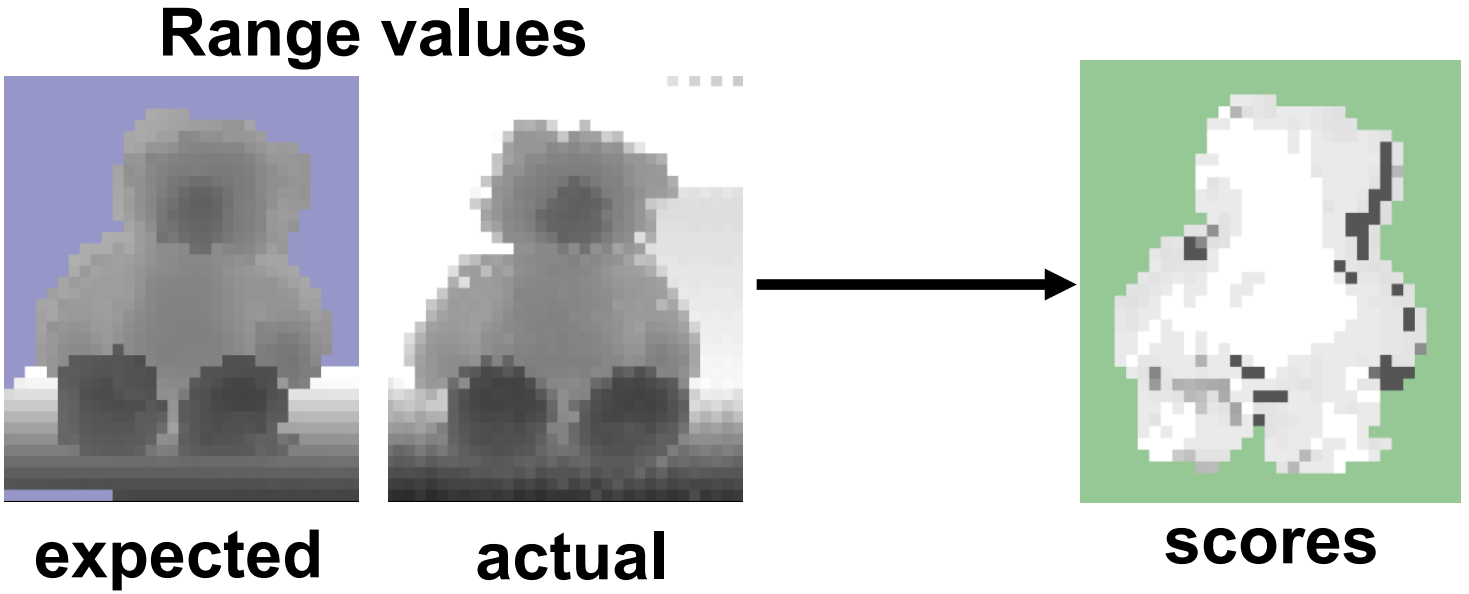
- Which poses fit the data best?
- Possible scores:
  - Sum of errors for nearest neighbors: For every s_{m_i}: Find nearest neighbor of $T \cdot m_i$ in the scene and sum up the distances:

$$\text{score} = \sum_i ||s_{m_i} - T \cdot m_i||$$

  - RANSAC on feature matches
  - …

Scoring for Range Images

- Score can be based on what we expect to see according to the model (simulation) and what actually has been perceived.



Calculation of Candidate Poses

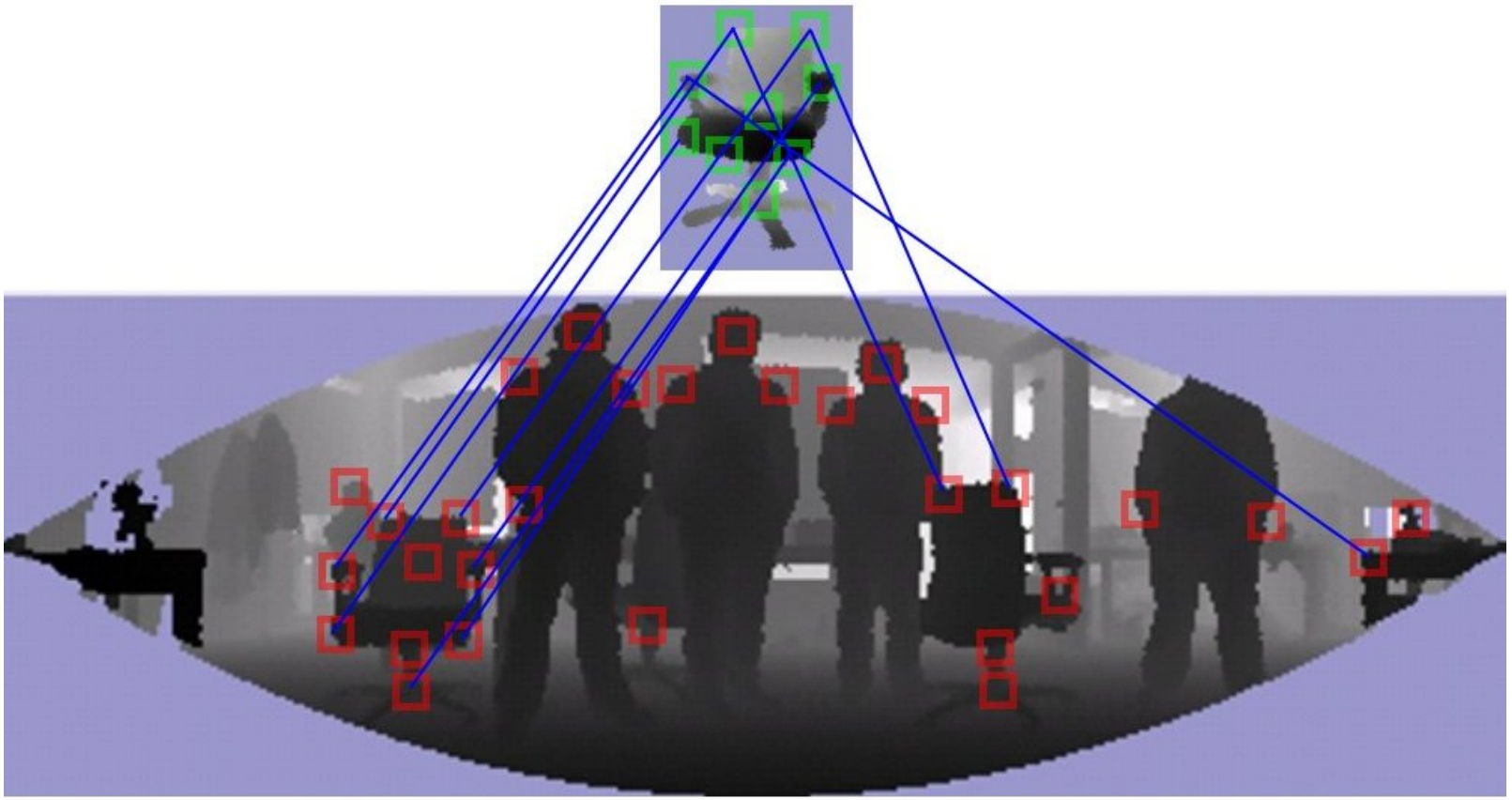
Theoretically, we could just check every possible pose according to a given discretization.

However, for six degrees of freedom (3D position + 3D orientation) this is too expensive!

Key question: How to get a **small number of candidate positions** that have a **high probability to include all true positions**?

Feature Matching

- Describe small parts of the scene in a way that makes fast comparison for similarity possible (extract features)

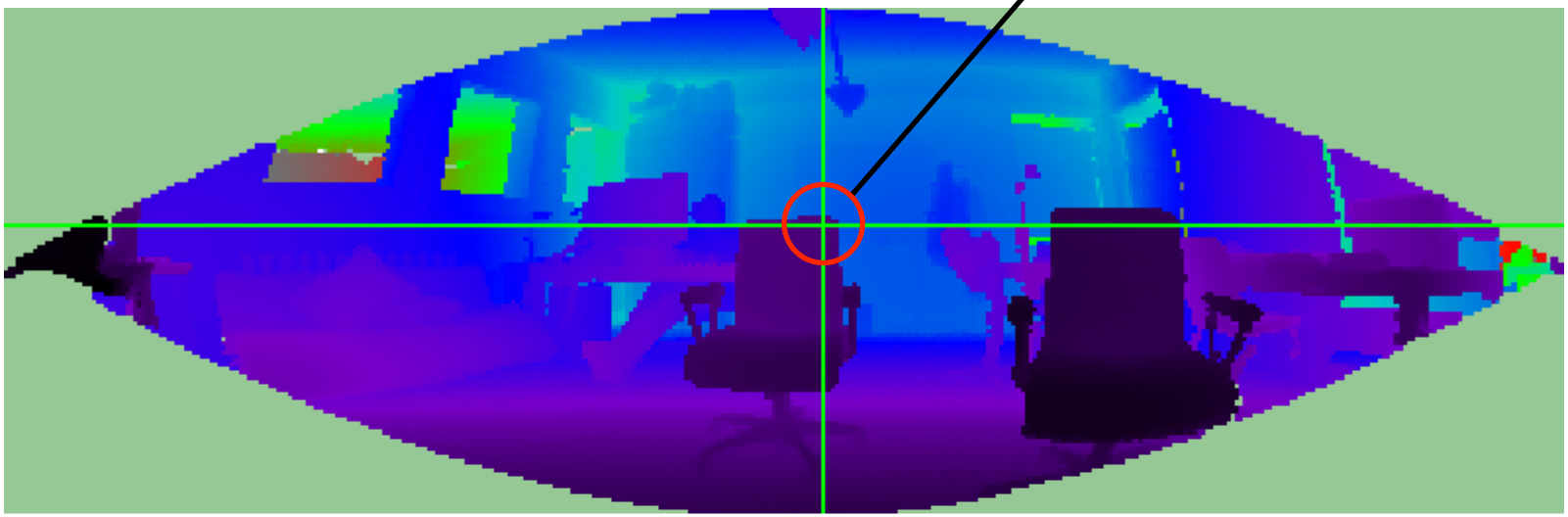


3D Feature Extraction

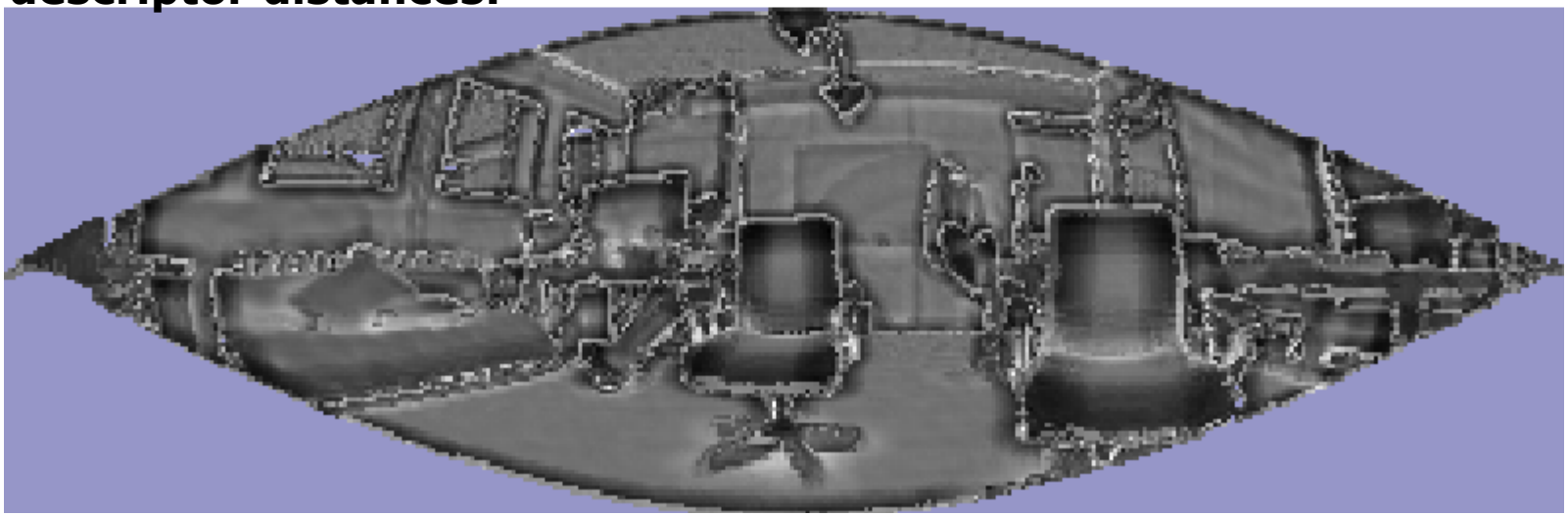
- Features like this typically provide what is called a *descriptor* or *description vector.* These are:
  - vectors of real numbers
  - comparable using standard norms/distances (Manhattan Distance, Euclidean Distance etc…)
  - the distance between two of the descriptors (*descriptor distance*) tells how different the corresponding areas are.
    (low distance means high similarity)

Descriptor Distances Corner

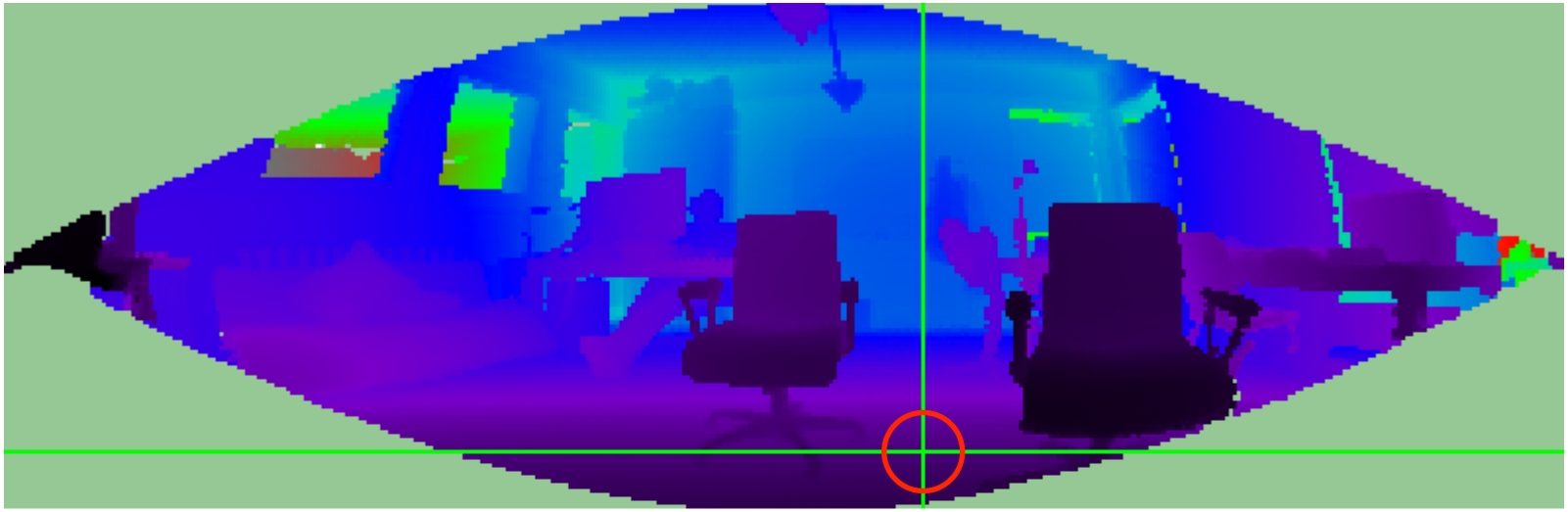
descriptor: (0.23, 0.13, 0.87, …)



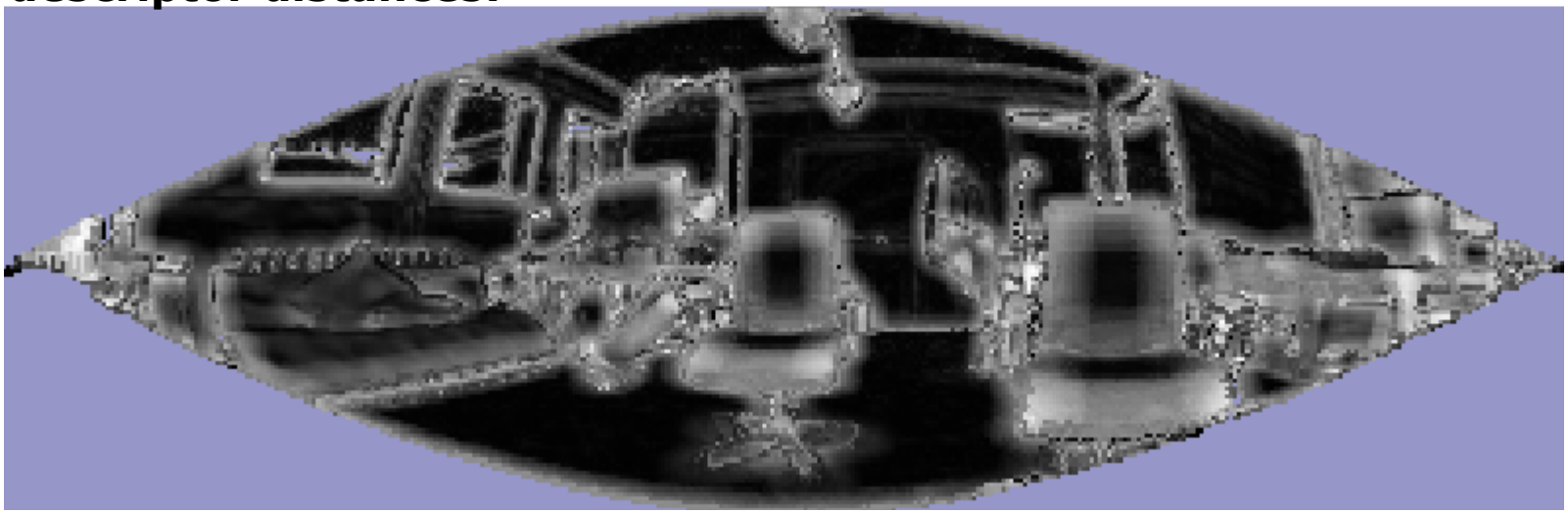
descriptor distances:



Descriptor Distances Planar



descriptor distances:



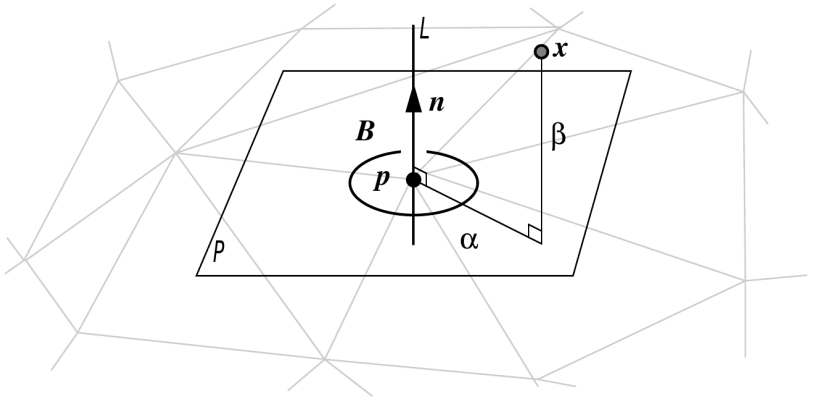
Calculate Pose from Matches

- As in ICP (Iterative Closest Points – see Introduction to Mobile Robotics), corresponding points (feature matches) can be used to calculate candidate transformations.

Example Features: Spin Images

- Describe the local environment of a point by mapping 3d points into a 2d histogram (i.e., a raster image)
- Robust w.r.t
  - small shape variations
  - occlusions
  - clutter
- Fast to compute – given an efficient data structure, e.g., a *k*d-tree

Spin Images

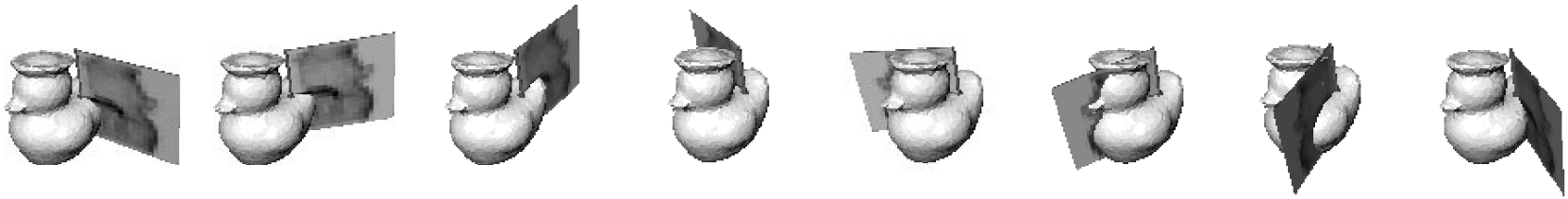


Mapping the 3d point $\boldsymbol{x}$ to the 2D histogram coordinate system:

$$S_0 : \boldsymbol{R}^3 \rightarrow \boldsymbol{R}^2$$

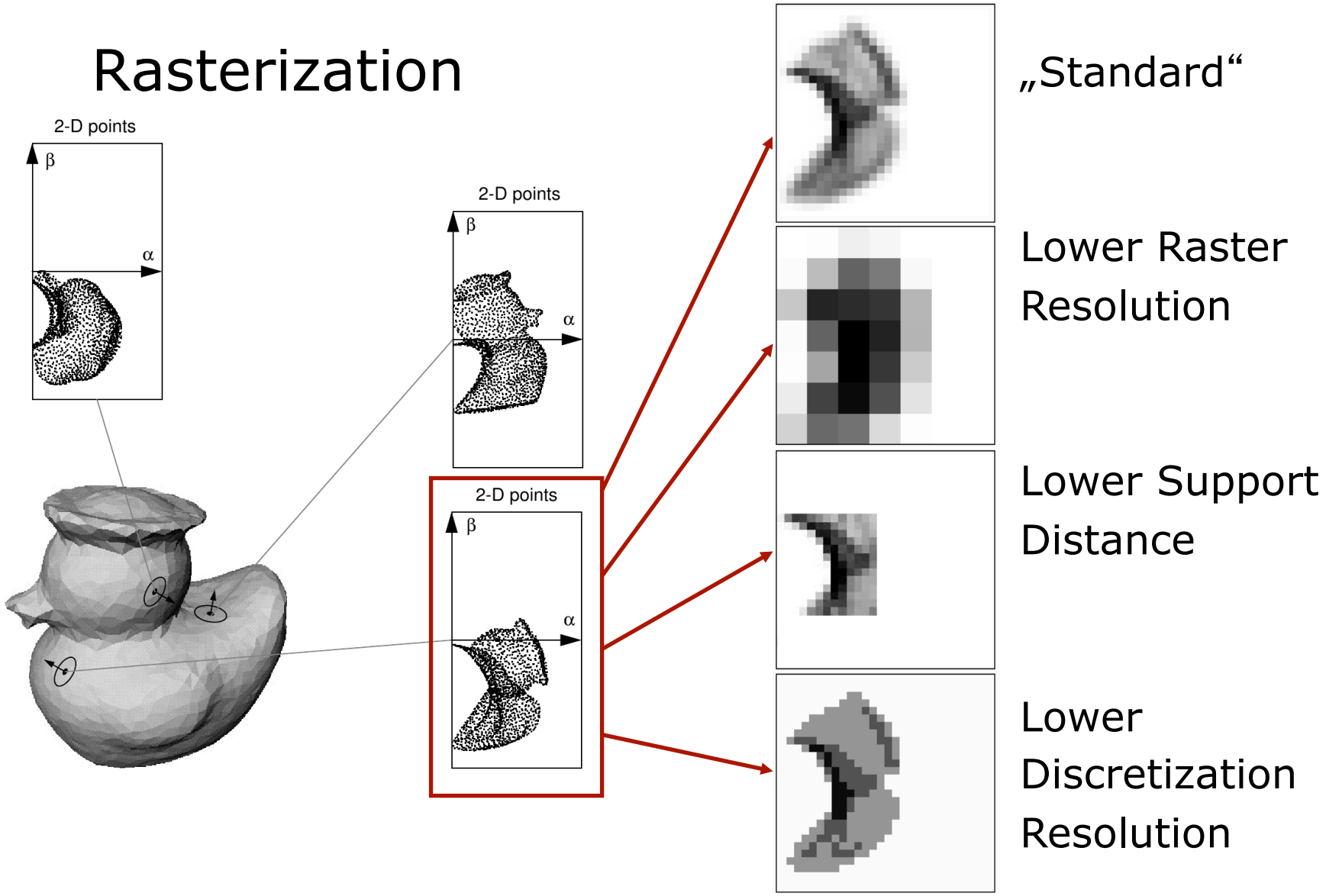
$$S_0(\boldsymbol{x}) \rightarrow (\alpha, \beta) = \left(\sqrt{\|\boldsymbol{x}-\boldsymbol{p}\|^2 - (\boldsymbol{n}\cdot(\boldsymbol{x}-\boldsymbol{p}))^2}, \boldsymbol{n}\cdot(\boldsymbol{x}-\boldsymbol{p})\right)$$

α and β are invariant w.r.t. the angle in the tangent plane. This mapping can be interpreted as „collecting“ points on a spinning image:



Spin Images

Rasterization



Where to extract the features?

- The remaining question is, where to extract the features:
  - **In every point in the point cloud**:
    Very expensive (extraction $O(n)$,
    matching $O(n^2)$)
  - **Randomly or uniformly sampled points**:
    Might miss important positions
  - ***Key point / interest point* extraction:**
    Try to find points that are in places that will be again detected in a similar scene.

Interest points in Vision

- For camera images there are a lot of standard methods for interest point extraction
  - The Harris Detector
  - SIFT interest point detection
  - SURF interest point detection
  - …
- They all work by analyzing the changes of the gradients in gray scale images and try to find places where those changes are unique and can be found again in slightly different views.

Interest points

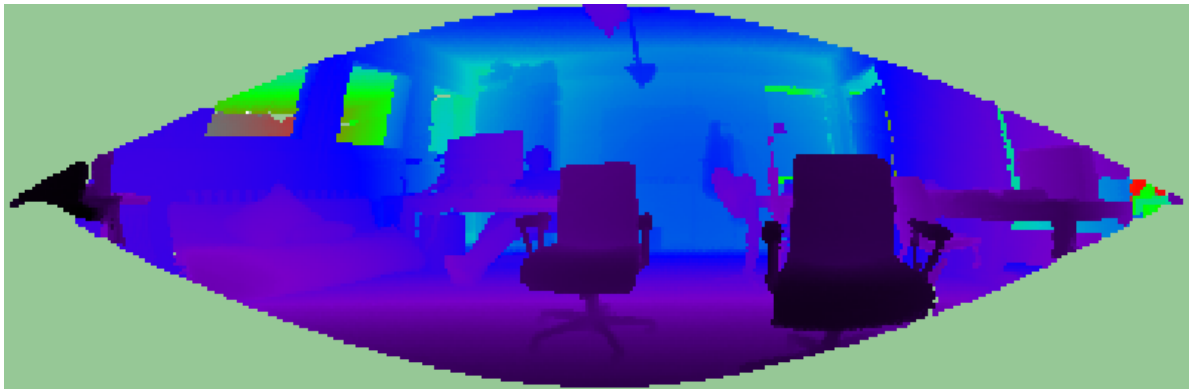
- In point clouds this is not such a well researched area, but the basic principles are similar:
  Analyze the surrounding of points and find stable positions that have
  - significant surface change
  - different dominant directions

  in the vicinity.

  The points are then typically determined using non-maximum suppression.

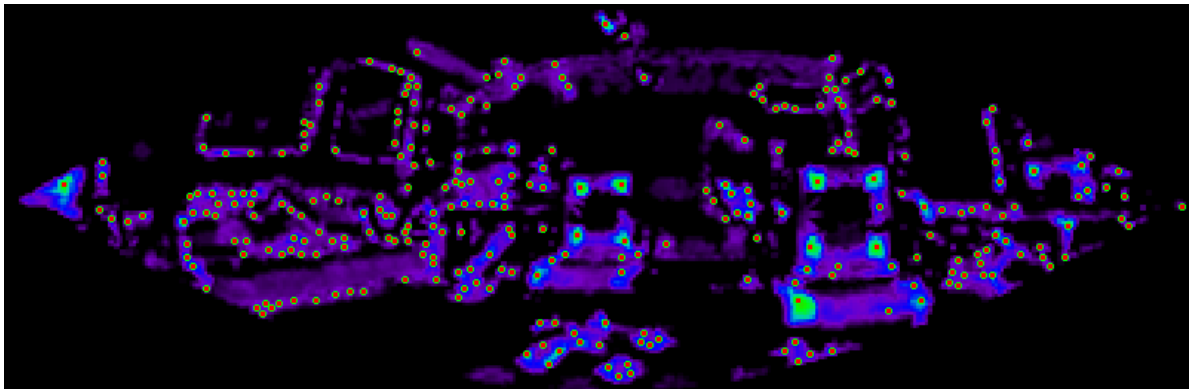
Key point extraction example

Scene



Surface change

Key points



Summary

- 3D Data are highly relevant for Navigation and Manipulation tasks
- There is a variety of approaches for sensing and representing 3D models
- Object recognition in 3D is carried out similar to how it is done in vision
- The features and interest points are not as sophisticated as in vision.