A Look into Mapping

“Simple” Mapping

given:

A) localization plus

B) environment sensing (e.g., free/occupied)

then:

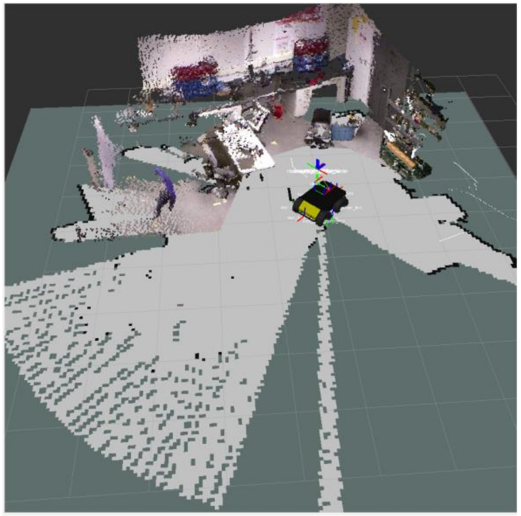
C) enter data into map representation

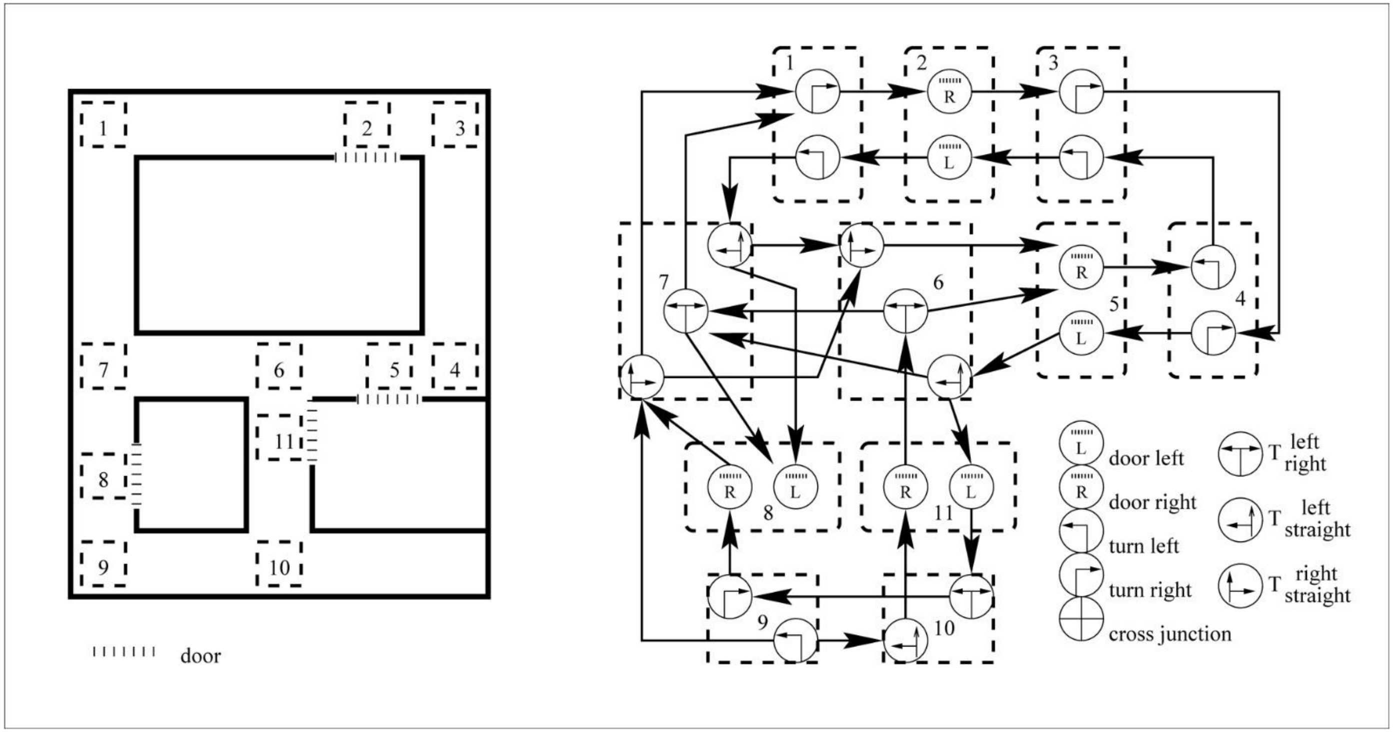
we now have a (first) look into C), then B)

Short Overview of
Map Representations

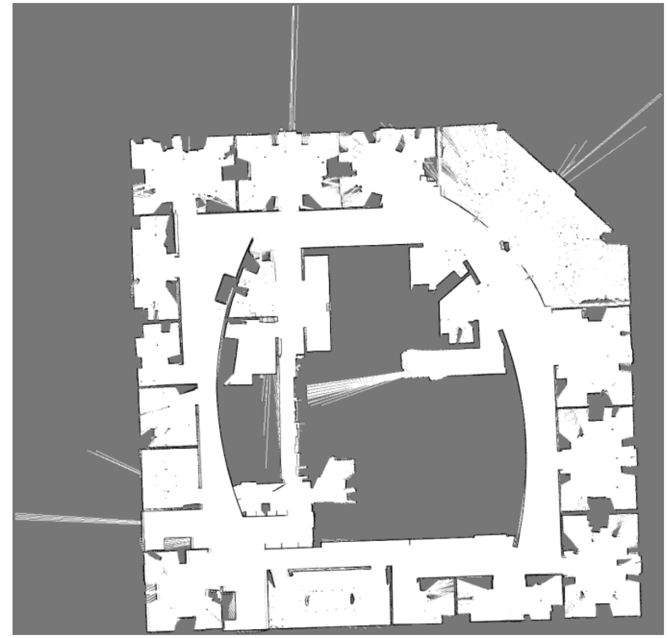
Maps in general

- geometric (Euclidean)
  - “proper” distances encoded
- topological
  - graph of “places” and “connections”





Metric: 2D/3D Occupancy Grid

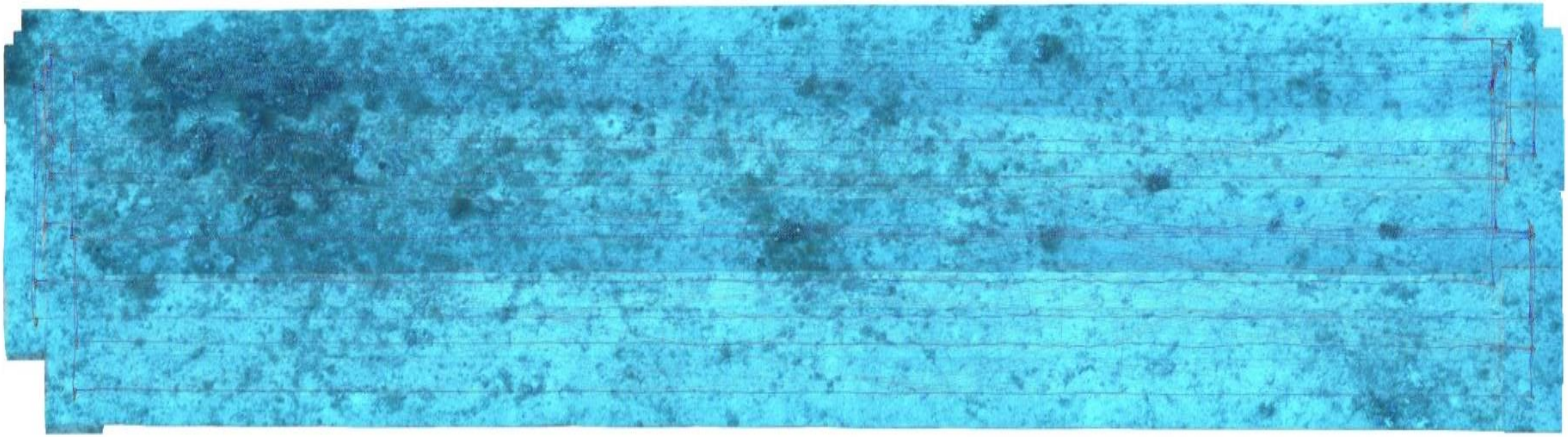


- 2D/3D regular grid (array)
  - with occupancy / free space / unknown info in each cell
  - x,y(,z) correspond to metric Cartesian coordinates
- simple but memory intensive
  - predominant representation for 2D
  - 3D less so, alternative: octtree

Metric: 2D Visual (Photo) Data

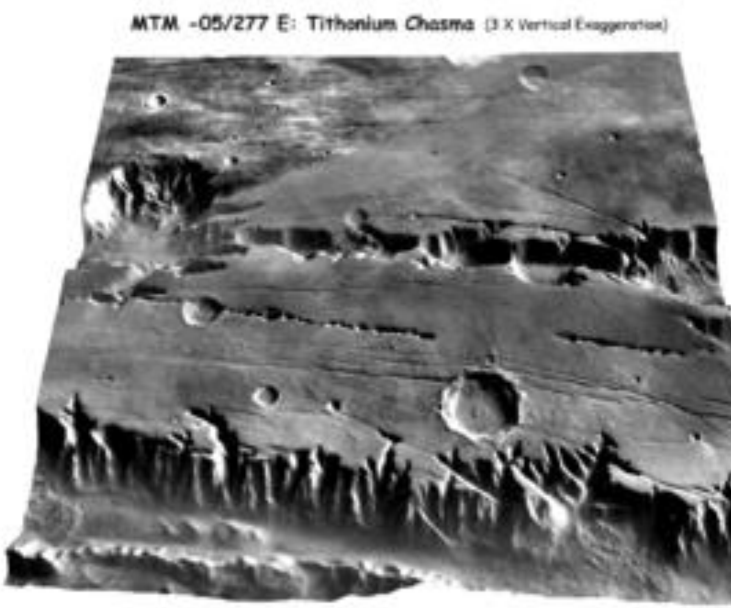
mosaic

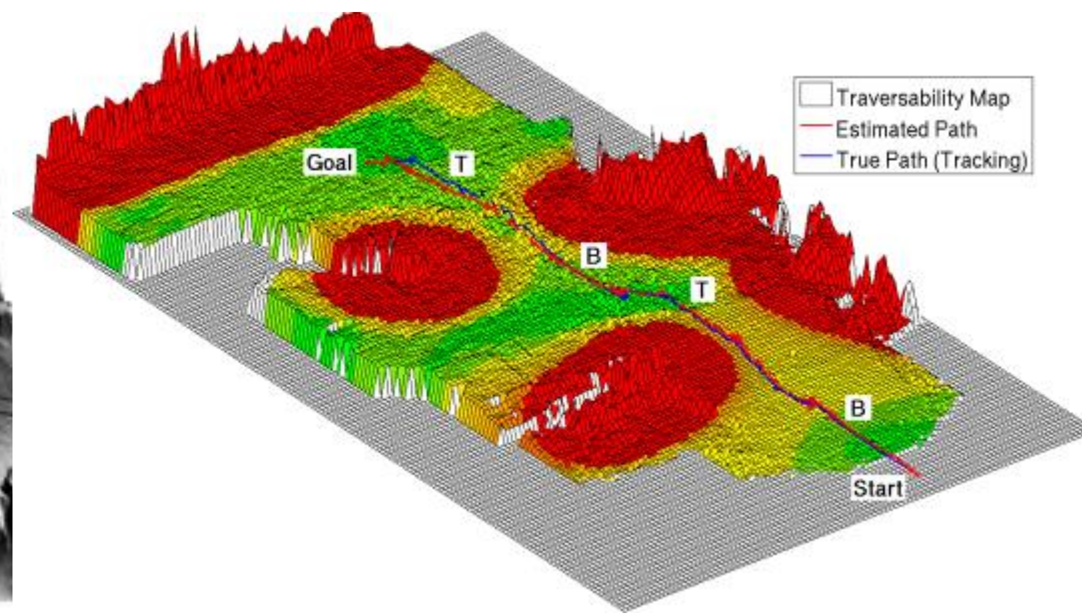
- 2D regular grid (array) with visual data in each cell aka **raster image**
- x,y correspond to metric Cartesian coordinates
- geo-referenced: use of geographical coordinates
  - gathered e.g. via GNSS
  - typically just for the (pose of) the origin
  - stored in image (meta-)data; see e.g. GeoPDF, GeoTIFF
- often (with meta-data) in Geo Information System (GIS)



(2.5 D) Elevation Map

- (typically regular) grid: value = elevation
- Digital Elevation Model (DEM),
  Digital Terrain Model (DTM),
  Digital Surface Model (DSM)
- often falsely denoted as 3D, but 2D manifolds in 3D space

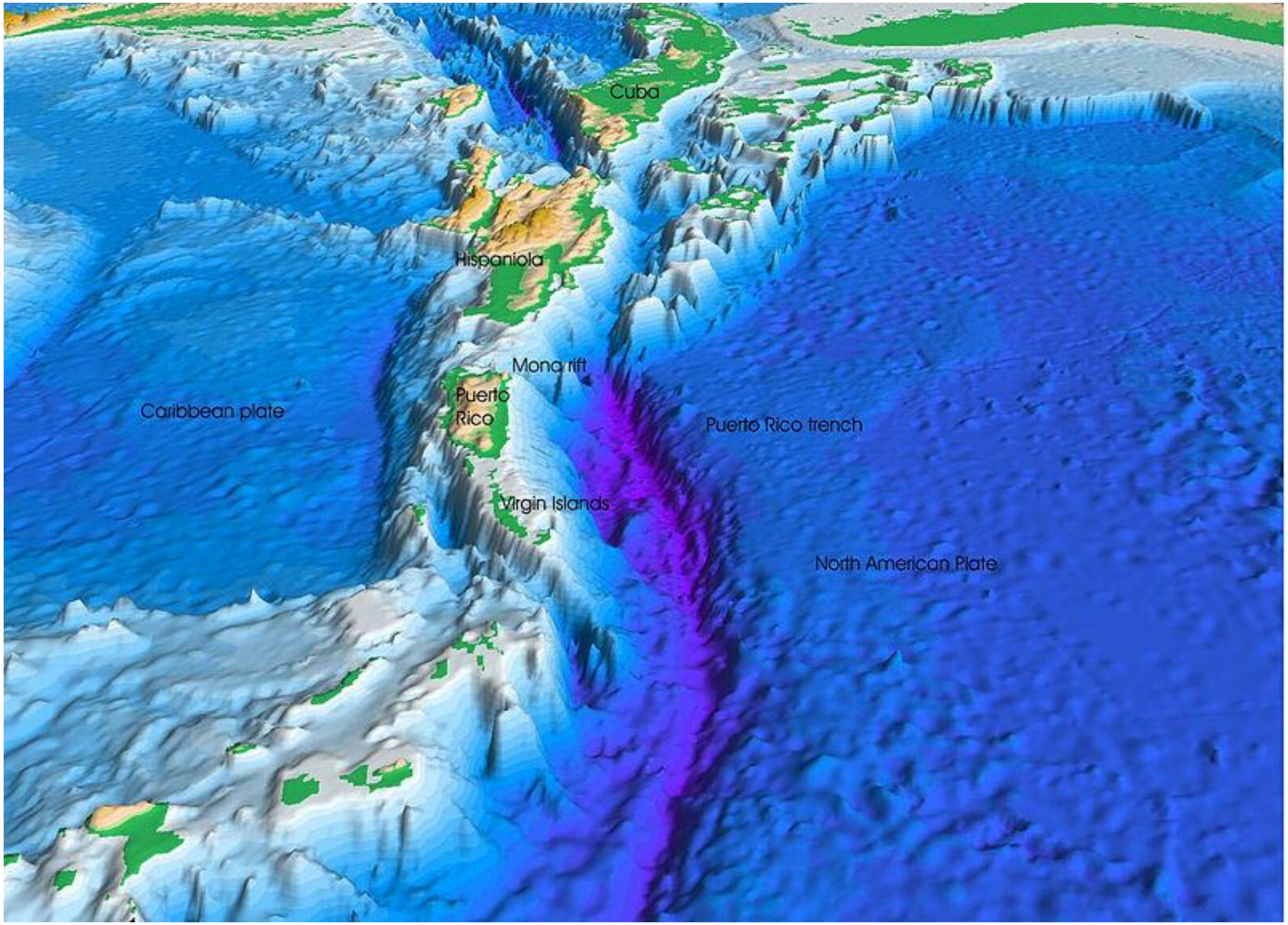




(2.5 D) Elevation Map

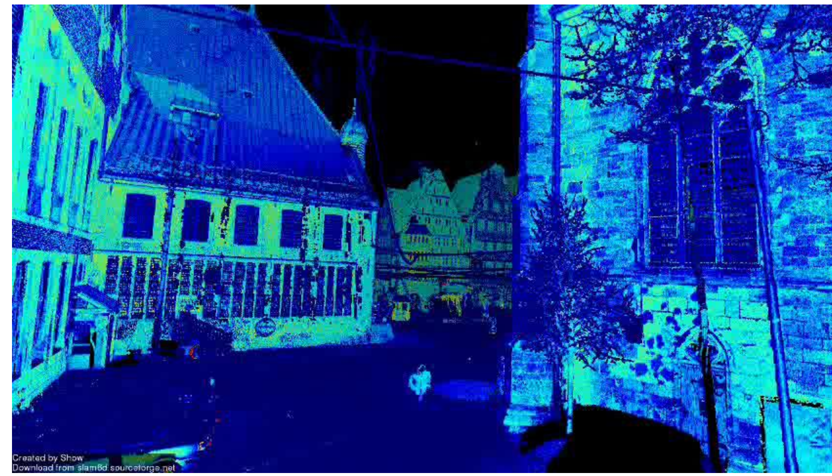
bathymetry

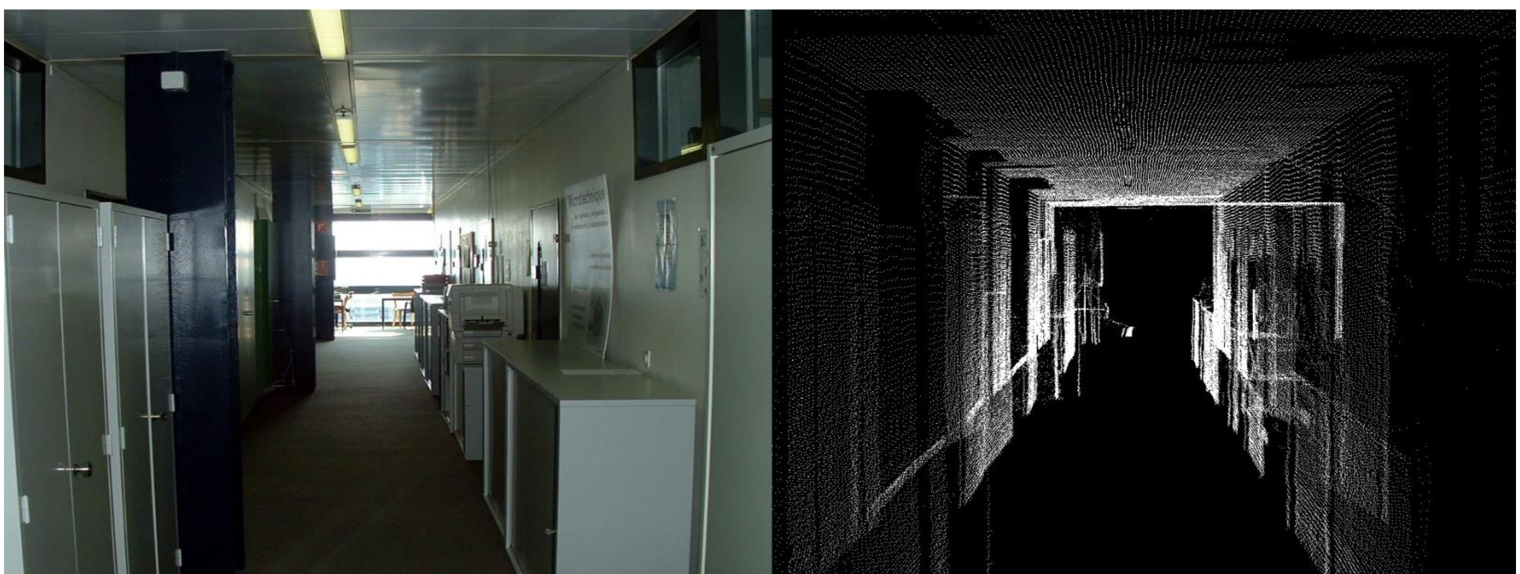
- underwater elevation map
- is **not** 3D



3D Point Cloud

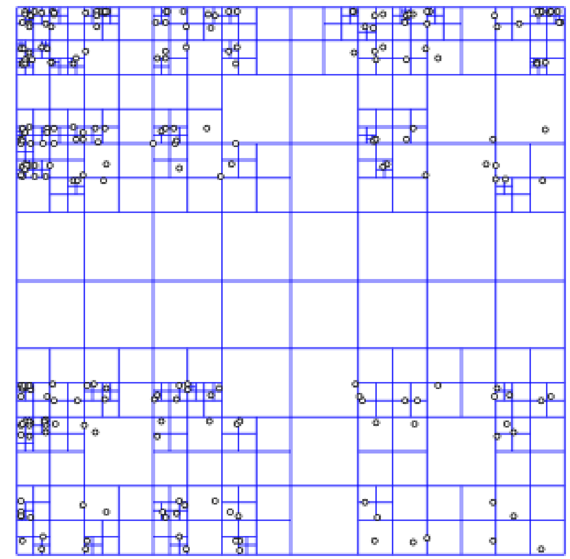
- set of 3D points
  - fine grain coordinates
  - point represents occupancy
- pro/cons
  - "raw" range data format
  - can grow very large
  - computational geometrie very hard



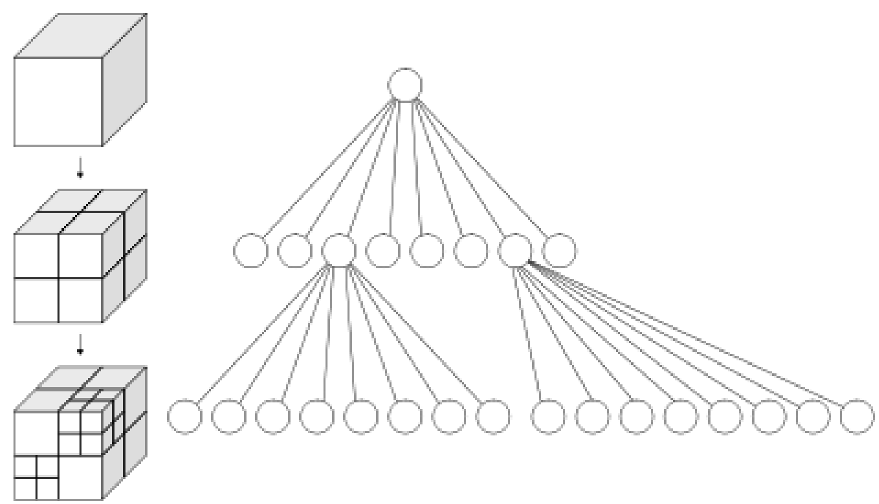


2D/3D Quad-/Octree

- volumetric representation
- recursive decomposition
  - divide in 4 / 8 cells
  - each occupied cell is further divided

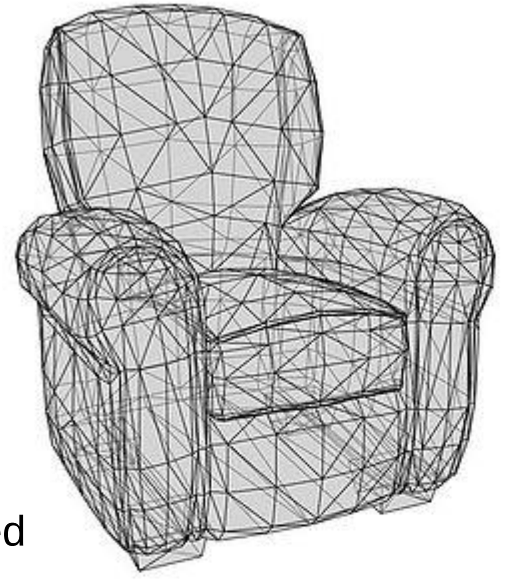


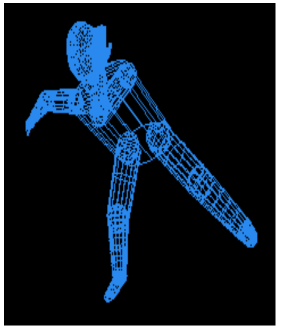
option for point clouds: store 3D info in cells

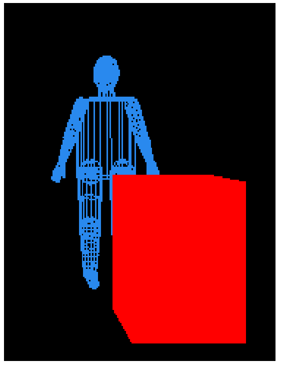


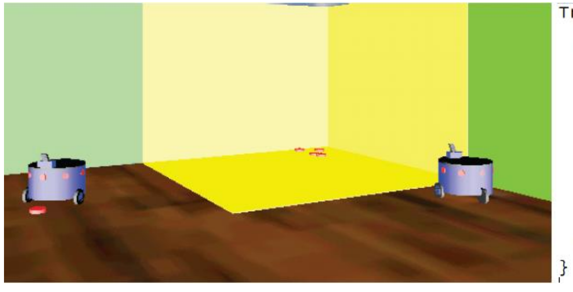
3D Surface Models

- in general
  - volume (voxel) vs surface representation
- surface
  - 3D meshes
    - regular polygons
    - points span surfaces
  - geometric object collections
    - planes, spheres, boxes
    - defined by parameters
- pros/cons
  - compact; relatively few data points needed
  - basis for computer graphics
  - basis for computational geometry
  - supported by standards (e.g., Collada)
  - difficult to extract from raw 3D sensor data









```
Transform {
  translation 0 0.15 0
  children [
    Shape {
      appearance Appearance {
        material Material  {
          emissiveColor  0 1 0
        }
      }
      geometry Cylinder {
          radius .2
          height .4
      }
    }
  ]
}
```

Representing Probabilistic Data: Evidence Grid Map

Conditional Probability

- probability of A under condition B

$$p(A|B) = \frac{p(A \cap B)}{p(B)}$$

- Bayes' theorem

$$p(A|B) = p(B|A)\,\frac{p(A)}{p(B)}$$

as $p(A \cap B) = p(B \cap A)$

Evidence Grid

grid map alg. (Moravec, Elfes, 80s)

- based on Bayes' theorem

$$p(A \mid B) = p(B \mid A) \frac{p(A)}{p(B)}$$

- using odds representation, p(A)/p(¬A)
  - o = cell occupied, s = sensor value
  - grid cell g(x,y): odds of being occupied

$$g(x, y) = \frac{p(o)}{p(\neg o)}$$

Evidence Grid

- we want:
  occupancy probability given sensor value $p(o \mid s)$
- core idea: use sensor model
  - likelihood of getting a sensor value depending on occupancy $p(s \mid o)$
  - can be derived a priori (through experiments)
- in combination with Bayes' theorem

same principle for free space ($\neg o$)

Incremental Update

- update: Bayes with sensor model

$$\frac{p(o\,|\,s)}{p(\neg o\,|\,s)} = \frac{p(s\,|\,o)\,\dfrac{p(o)}{p(s)}}{p(s\,|\,\neg o)\,\dfrac{p(\neg o)}{p(s)}} = \frac{p(s\,|\,o)\,p(o)}{p(s\,|\,\neg o)\,p(\neg o)} = \frac{p(s\,|\,o)}{p(s\,|\,\neg o)} \cdot \frac{p(o)}{p(\neg o)}$$

- assume independence of measurements

$$p((o\,|\,s_t) \wedge (o\,|\,s_{t-1})) = p(o\,|\,s_t) \cdot p(o\,|\,s_{t-1})$$
$$p((\neg o\,|\,s_t) \wedge (\neg o\,|\,s_{t-1})) = p(\neg o\,|\,s_t) \cdot p(\neg o\,|\,s_{t-1})$$

$$g(x,y)_t = \frac{p(s\,|\,o)}{p(s\,|\,\neg o)}\, g(x,y)_{t-1}$$

Log Odds for Efficiency

$$g'(x,y) = \log\left(\frac{p(o)}{p(\neg o)}\right) = \log\big(p(o)\big) - \log\big(p(\neg o)\big)$$

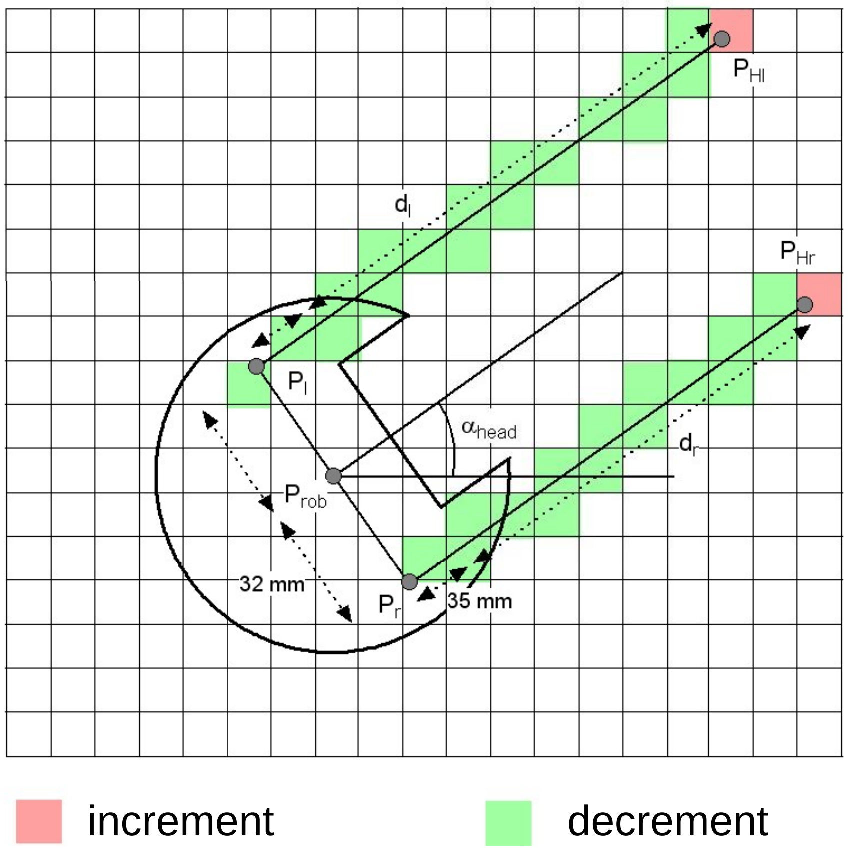
- multiplication becomes addition
- simple integer representation (e.g., byte)
- initialization p(o) = p(¬o) = 0.5 ⇔ g'(x,y) = 0
- sign indicates occupancy, value confidence

Need for sensor models

narrow beam range

- e.g., laser range finder

- here: two aIR sensors
- Bresenham line drawing
  - open space (line): add v1
  - end point: add v2
  - v1, v2 = log(p(s|o)/p(s|¬o))
  - naive: v1, v2 = -1 , +1

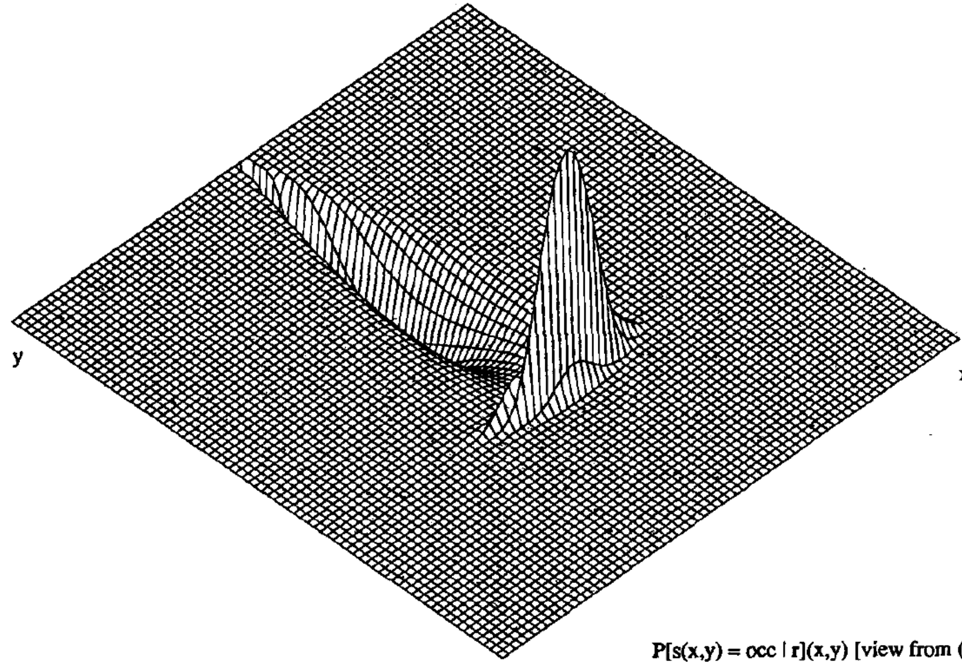


Need for sensor models

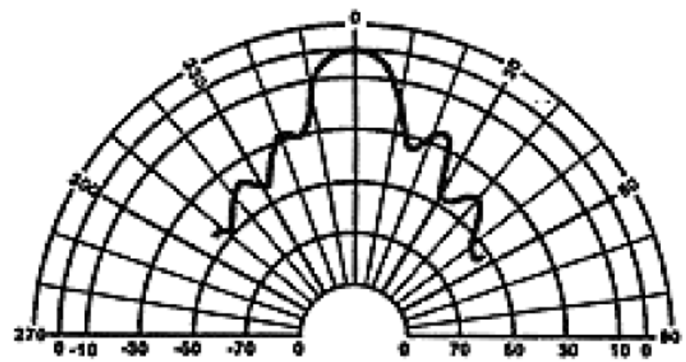
wide beam range

- e.g., ultrasound

- algorithmic approach
  - inconvenient
  - complex distributions
- store profiles
  - lookup tables



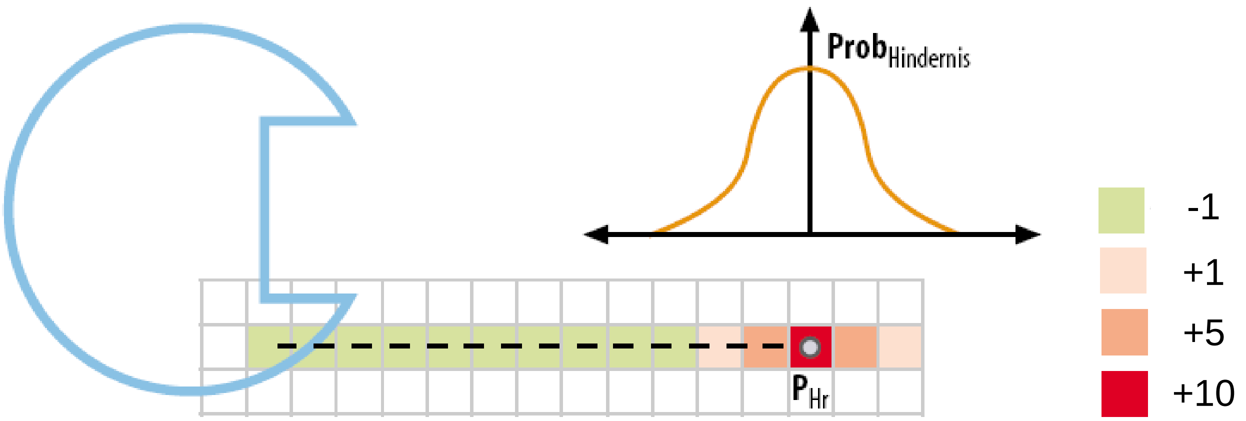
P[s(x,y) = occ | r](x,y) [view from (0,0)]



Typical Beam Pattern At 50 kHz

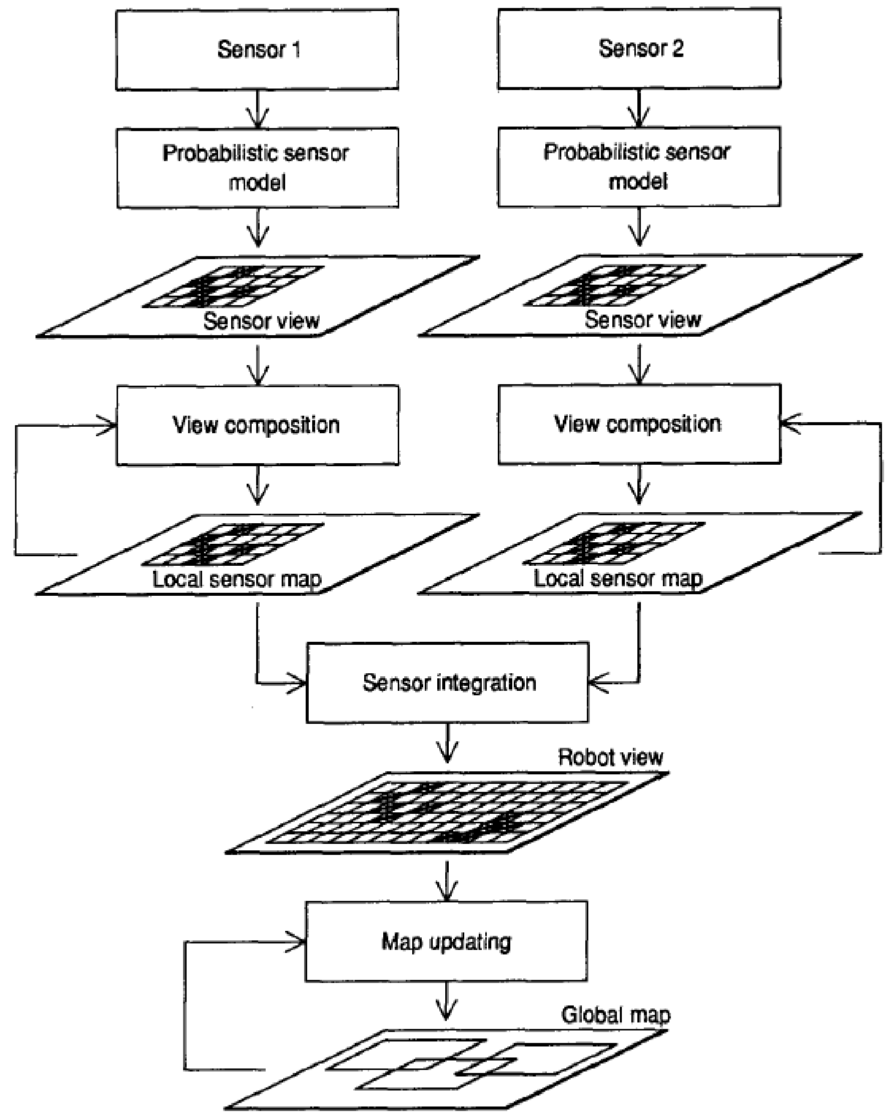
Sensor Models

- unlimited possibilities for modeling
- in theory
  - for all sensor values s
  - distribution of probabilities
  - for all positions x,y (from sensor to global frame) needed
- often in reality
  - simplified coverage and probabilities
  - algorithmic model (e.g. narrow beam) or look up (e.g. ultrasound)



Sensor Fusion

- sensor models
  - data in local grids
- can easily be fused
  - log odds => addition



Range Sensors

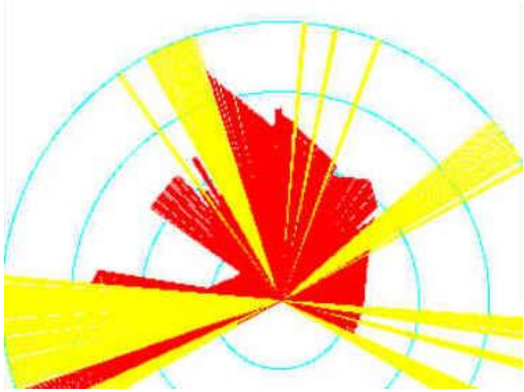
(more precisely: distance to obstacles)

Laser Range Finder (LRF)

- aka laser scanner, laser/light radar (ladar/lidar)
- time of flight: speed of light is very fast
  - timing with digital electronics challenging
    - e.g. capacitive discharge for time-measurements
    - interference (slow amplitude modulation)
  - pulsed versions have high sampling-rate
    - rotating mirror: high spatial resolution
- very accurate

LRF Example

- Hokuyo URG-04LX
  - field of view: 240°
  - angular resolution: 0.36°
  - response time: 100 ms
  - resolution: 20 mm
  - range: 4 m
  - power: 2.5 W at 5V
  - weight: 0.16 kg
  - dimensions: 50 x 50 x 70 mm (L x W x H)

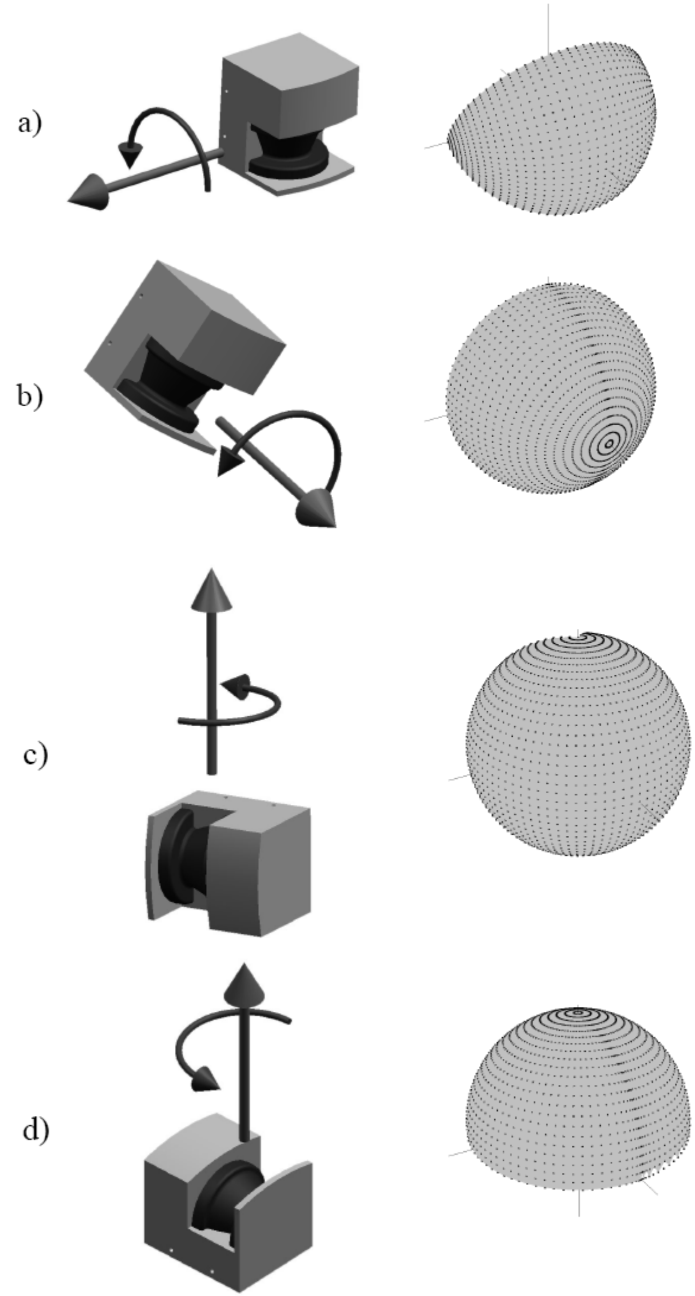


LRF Example

- SICK LMS 200
  - field of view: 180°
  - angular resolution: 1 - 0,25°
  - response time: 13 - 53 ms
  - resolution: 10 mm
  - error: +/- 15 mm
  - range: 80 m
  - power: 20 W at 24V
  - weight: 4,5 kg
  - dimensions: 156 x 155 x 210 mm (L x W x H)

Actuated LRF

- standard LRF: 2D
  - horizontal scan plane
- actuated => 3D
  - servo motor
  - 1 DOF rotation
  - possible movements
    - pitch
    - roll
    - yaw side
    - yaw up



High-End 3D LRF examples

- FARO Focus 3D
  - ~1 million points / sec (color CCD included)
  - 120 m range
  - FOV: 360 x 305 deg

- RIEGL VZ-400
  - 120,000 points / sec
  - 600 m range
  - FOV: 360 x 100 deg

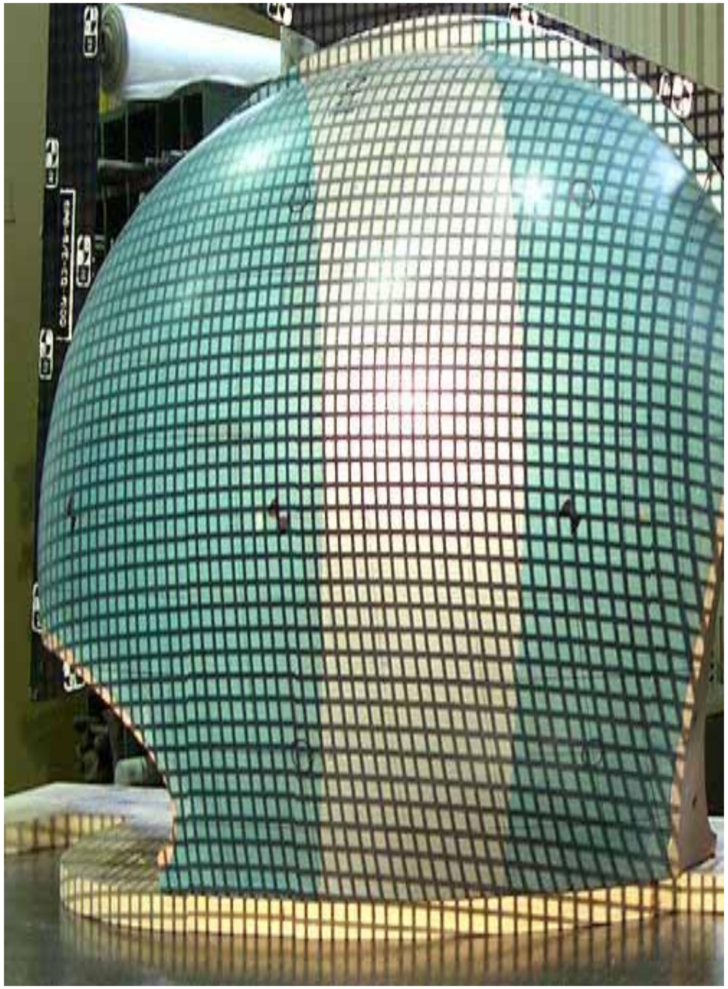
- Velodyne HDL-32E
  - 0.7 million points / sec
  - 32 laser elements
  - FOV: 360 x 40 deg (+10 to -30 deg)
  - 100 m range (+/- 2cm)
  - compact, light, low-power

Structured Light

pattern of light

- with know properties
- typical: grid or pseudo-random

- derive shape
- from the geometric deformations



Structured Light: Kinect

- inexpensive game controller (gestures)
- for MS Xbox 360
  - active IR pattern and sensor
  - plus color camera
  - aka RGBD (color + depth)
- caveats
  - does not work outdoors (IR)
  - field of view and range OK, but…
  - noise and resolution OK, but…

Ultrasound (US) Sensors

- relatively low cost sensors
- field of view
  - often broad (60-120°)
  - narrow ones exist (10-20°)
- limitations
  - speed of sound in air is slow: ~ 300 m / s
  - and depends on humidity, temperature, ...
  - need to handle echos, multiple paths

US Example

Baumer UNAM 30U6103

- range: 10-70 cm
- resolution: 0.3 mm
- field of view: 10 deg
- power: <1W at 15-30V
- linear analog output 0-10V

typical obstacle sensor,

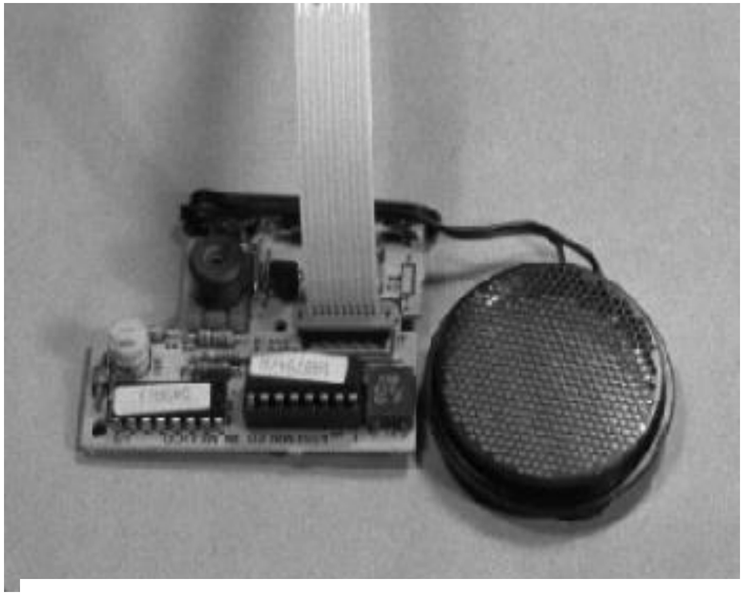
respectively simple „object“ sensor in automation

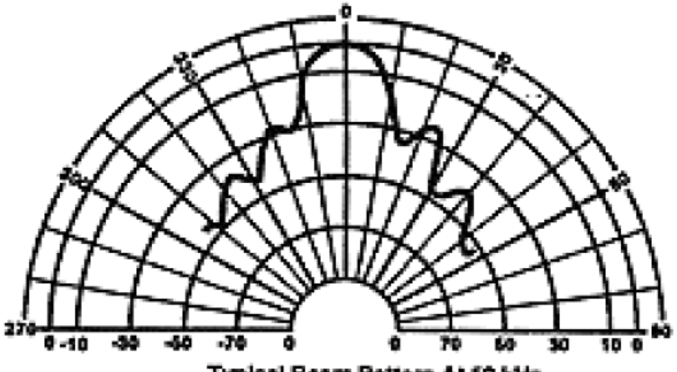
US Example

Polaroid 6500 & 600

- ranging board 6500
- transducer 600

- range: up to 10m
- field of view: 120 deg
- power: 5V
  - 2 A during transmit
  - 0.1 A else





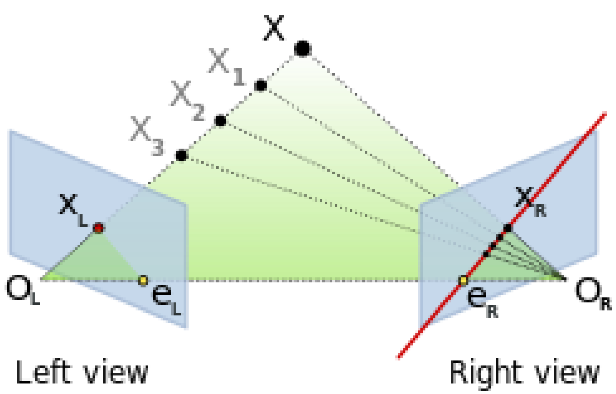
Typical Beam Pattern At 50 kHz

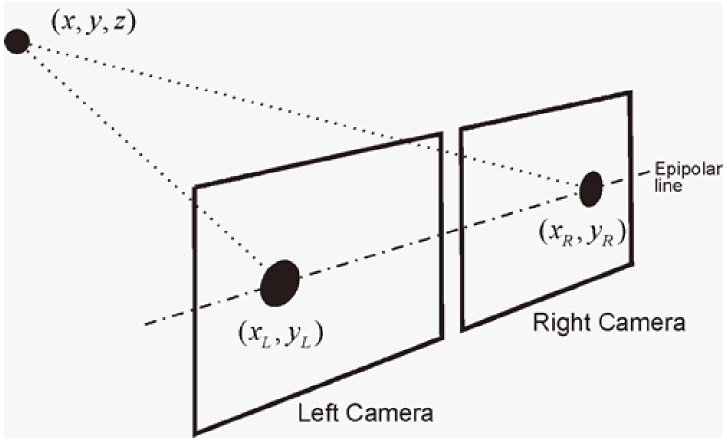
Passive Range Sensing

(i.e., vision based)

(Glimpse into) Stereo Vision

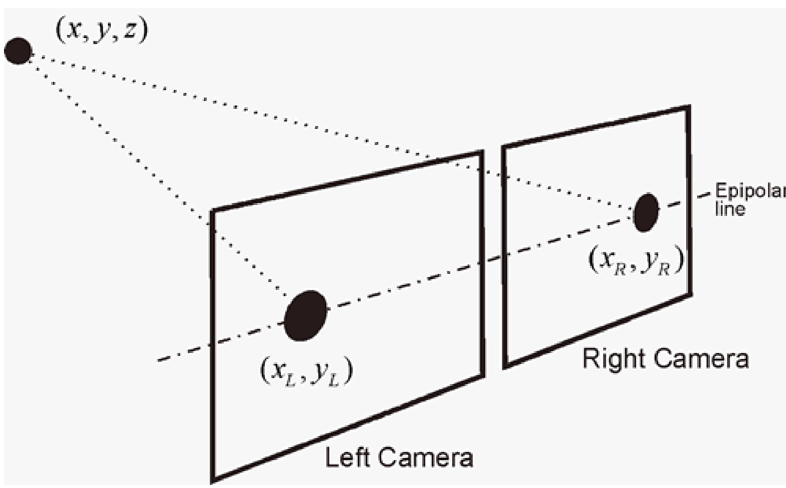
- rectified images
  - i.e., no camera distortions and on one plane (to ease search for correspondences: horizontal epipolar line)
- epipolar geometry
  - corresponding projected points (aka epipoles)
  - and center of projection
  - are on a line (aka epipolar line)

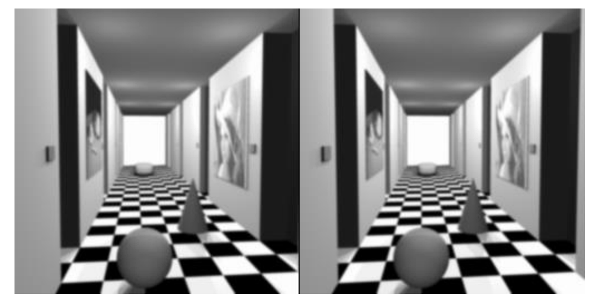


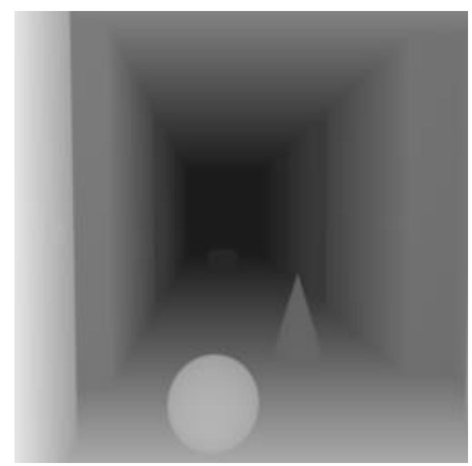


Stereo Vision

- epipolar line
  - corresponding points *p*, *p'* on a horizontal line
  - distance |*p*, *p'*| aka disparity plus triangulation
  - gives range, respectively full 3D coordinates







Stereo Vision

- note: stereo typically used as 2.5D sensor
  - i.e., ***dense*** feature/range measurements
  - no need for interest points: brute force try all
  - computationally feasible due to epipolar constraint
  - and simple matching (e.g., pixel blocks)

- occasionally (combined with): ***sparse*** stereo
  - get (few) but very robust 3D landmarks
  - to match (3D-register) them across several stereo-pairs (visual odometry or sparse 3D mapping)
  - then typical interest-points & descriptors used (e.g., Harris & SIFT)

(a very short glimpse into) Simultaneous Localization and Mapping (SLAM)

SLAM

chicken & egg problem

- build map (needs localization)
- while using map for localization

main idea

- exploit spatial relations to previously visited places (loop closing)
- to bound the cumulative error

SLAM

usually treated as two parts

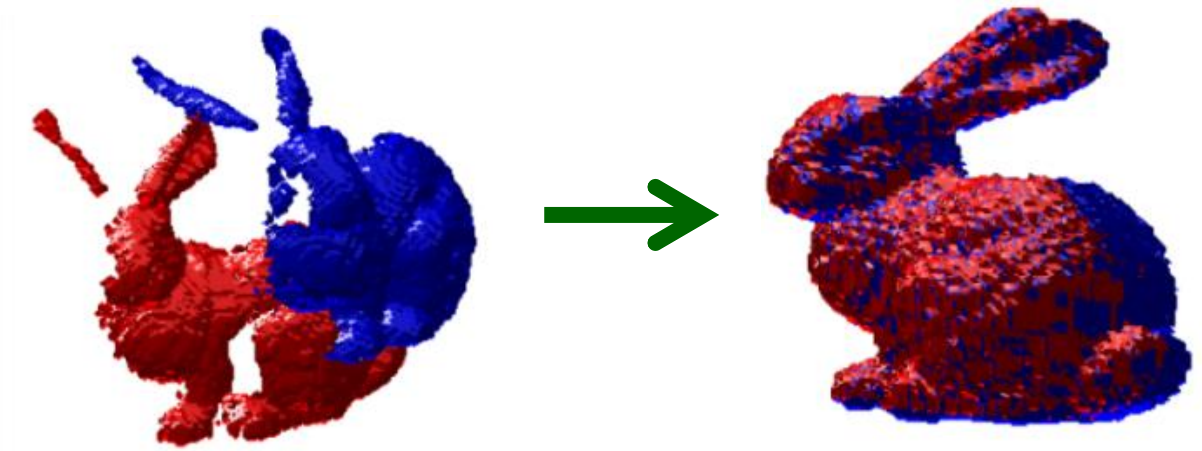
- front-end
  - 2D/3D registration (and odometry)
  - plus related uncertainties as weights
  - sequentially plus loop-closures
- back-end
  - generic optimization of the weighted constraints

Relative Pose Estimation via Sensor Data Registration

Sensor Data Registration

problem:

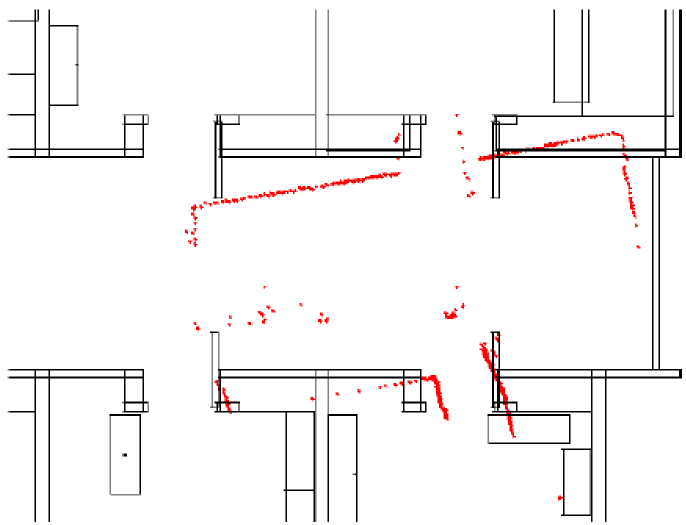
- given two sensor data sets
- find parameters of transforms to spatially align them,
- i.e., 3-DoF (2D), 4-DoF (2.5D) or 6-DoF (3D)
- incl. uncertainty estimates for prob. localization & SLAM



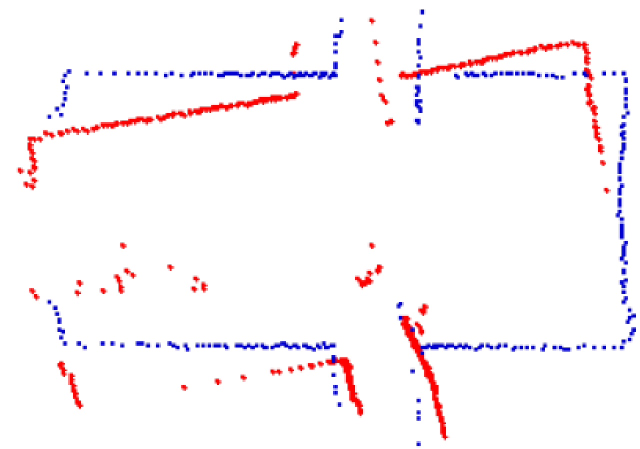
Example: Scan Matching

based on LRF data

match to



vector map

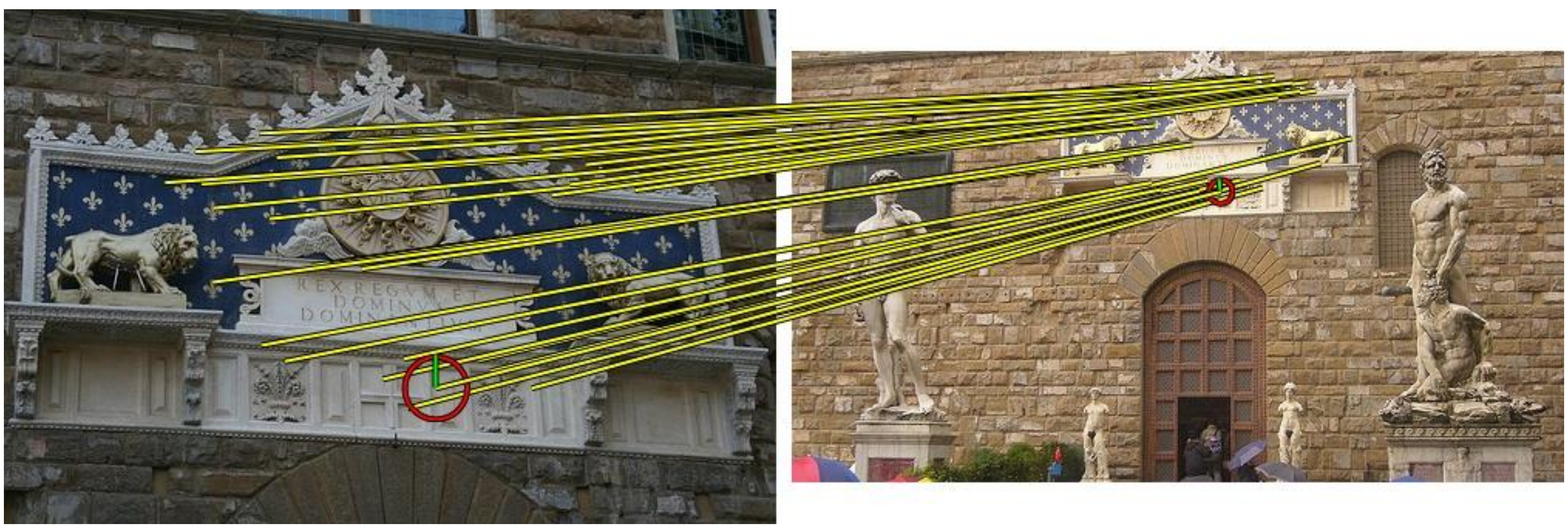


grid (scan or map)

Example: image registration

e.g., using visual **features**

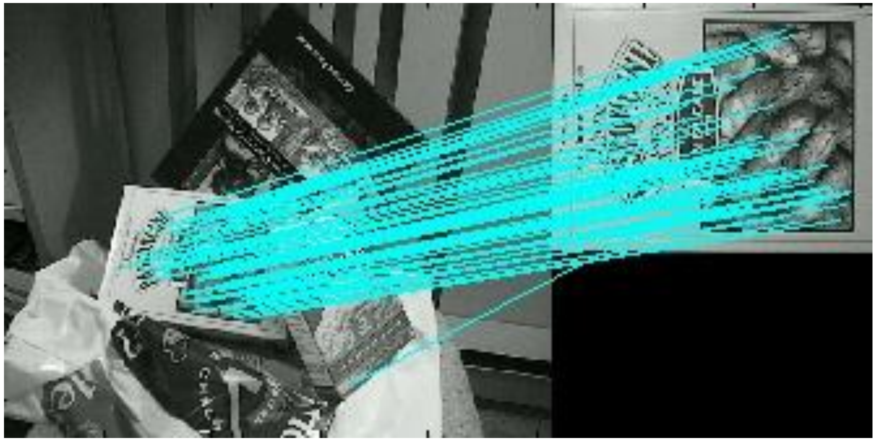
- **interest points**
  - to get a feasible subset of the whole image
- **descriptors** at the interest points
  - to solve the correspondence problem



(Natural) Image Features

visual features

- interest points
  - do not consider all pixels
  - but find "distinctive" locations
- descriptors
  - generate "unique" representation of the location
  - to allow matching across different images



very active field in computer vision

- for stereo, registration, ...
- speed / robustness trade-off very important

popular robotics combination:

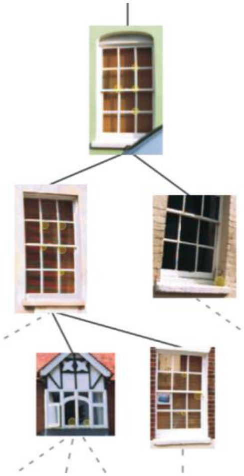
- Harris corner (interest)
- SIFT or SURF (descriptor)

Loop Closing

Loop Closing

- proximity based
  - easiest possible strategy: use current localization estimate
  - to check whether there are previously visited places around
  - if so: try registrations
- place recognition
  - non-trivial, especially with respect to good strategies to get reasonable cost/benefit
  - typically (visual or 3D) feature collections in associative representation (hashes)
  - e.g., by using FABMAP in the context of metric SLAM

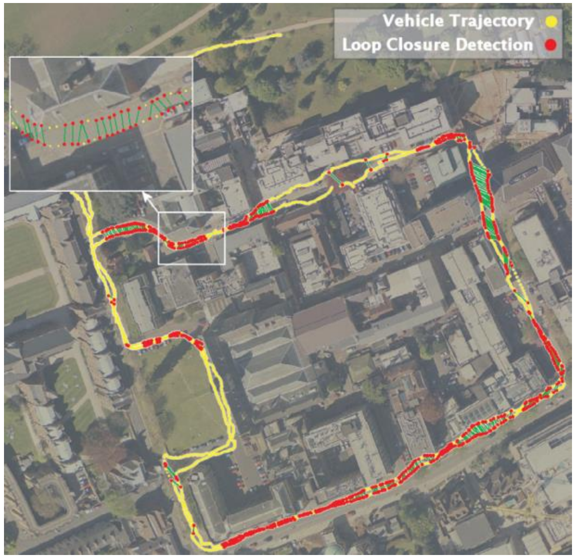
Appearance based Mapping: FAB-MAP



- learn a generative model of place appearance
- based on bag of words on SURF

very efficient

- sub-linear in the number of places



SLAM backend

SLAM backend:
Kalman filter

Kalman Filter for SLAM

- state vector robot motion in 2D
  - state vector is 3x1: [x,y,theta]
  - covariance matrix is 3x3
- mobile robot kinematics are not linear
  - use of EKF (or better UKF)
- SLAM: state vector is expanded
  - to include landmark positions
  - covariance matrix is also expanded

$$X = \begin{bmatrix} X_R^T & X_{L_1}^T & \dots & X_{L_n}^T \end{bmatrix}^T$$

$$\begin{bmatrix} P_{RR} & P_{RL_1} & \cdots & P_{RL_N} \\ P_{L_1R} & P_{L_1L_1} & \cdots & P_{L_1L_N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{L_NR} & P_{L_NL_1} & \cdots & P_{L_NL_N} \end{bmatrix}$$

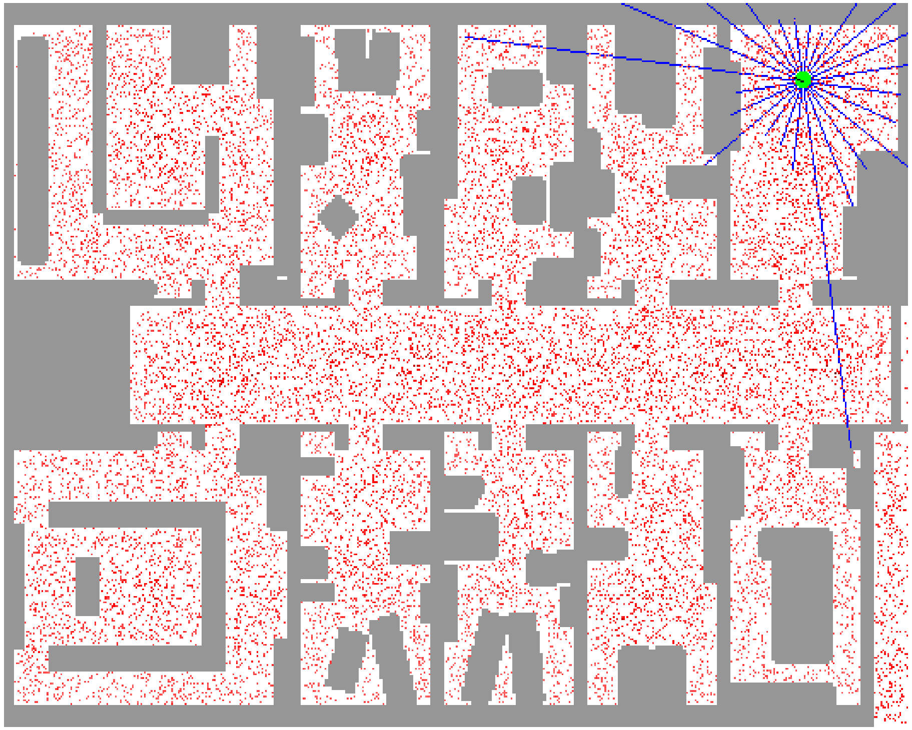
Kalman Filter for SLAM

- challenges for Kalman Filter SLAM
  - n landmarks => n^2 variables in covariance
  - hence computationally challenging
    (can be partially mitigated using sparse matrix algebra)
  - EKF tends to be unstable (hence option UKF)
- alternatives are hence popular
  - e.g., particle filter
  - e.g., graph based

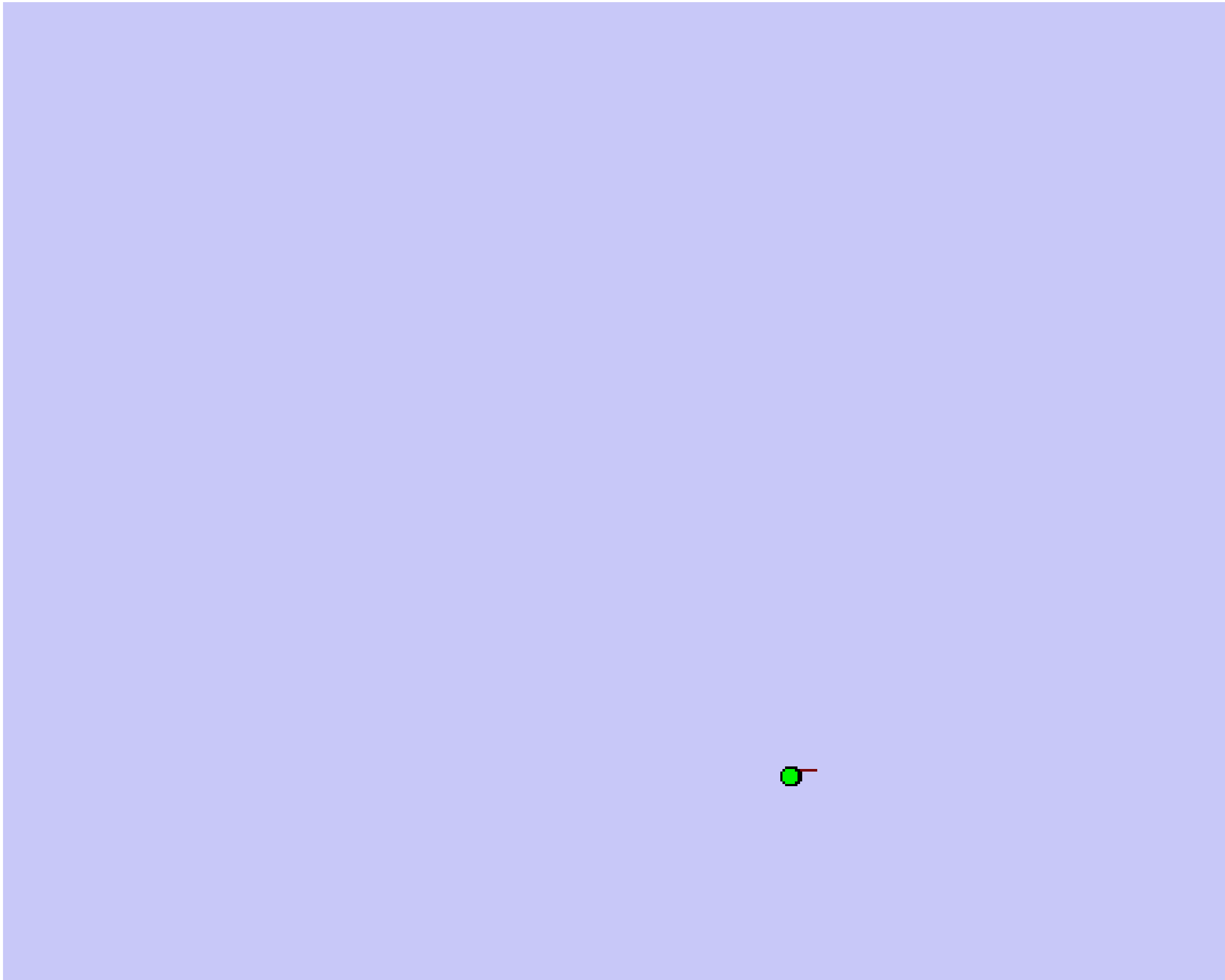
SLAM backend: particle filter

Particle Filter

- represent distribution
  - by samples (particles)
  - population based somewhat similar to Evolutionary Algorithm
- (recap) example: localization
  - (from Dieter Fox, UWash)
  - 24 sonar sensors
  - robot drawn at estimated position,
  - which is not the correct one in the beginning



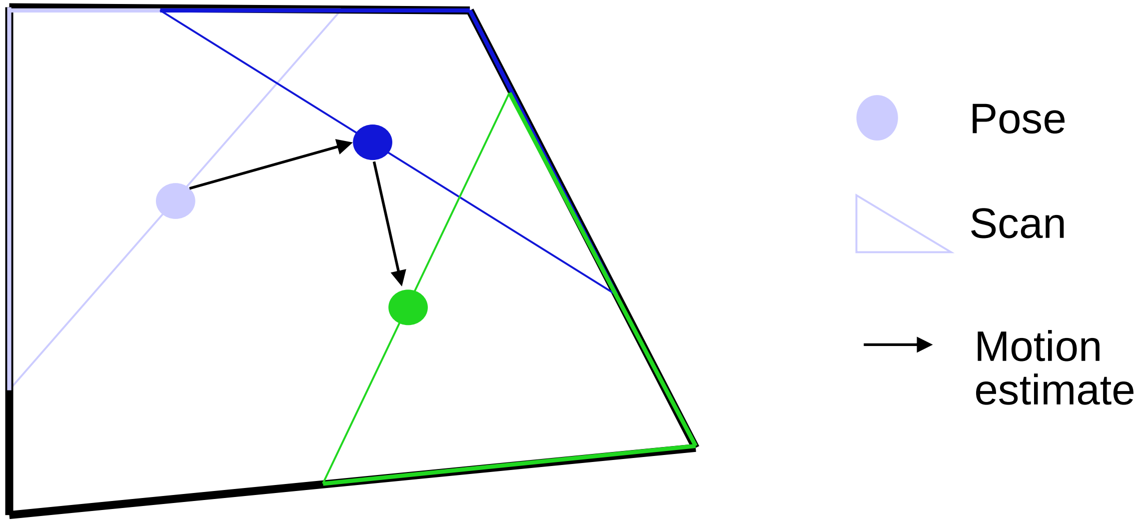
Example Mapping



SLAM backend:
pose graph

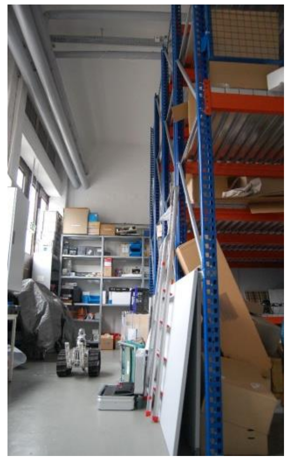
Pose Graph

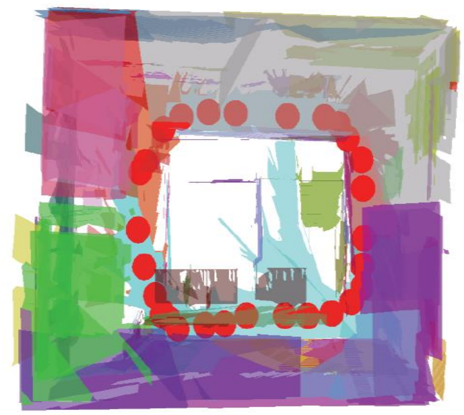
- nodes: sensor observations
  - e.g., 2D/3D scans, images
- edges: pose difference estimates with uncertainties
  - motion estimates via odometry and/or registration(s)
  - introduce constraints

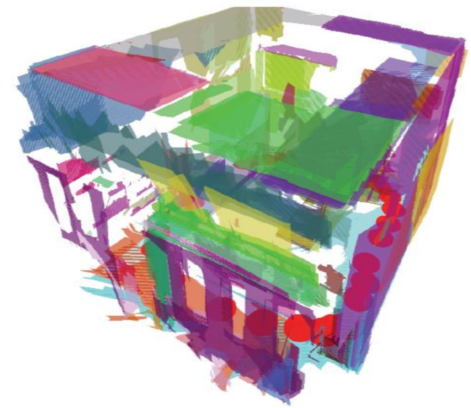


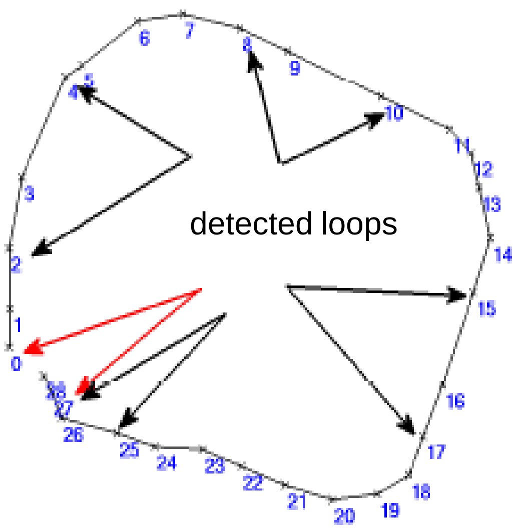
Loop Closing

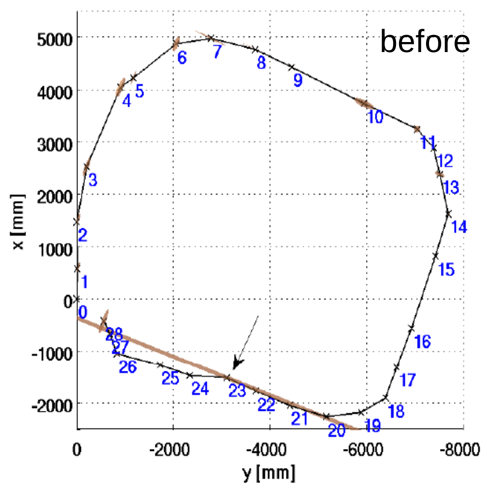
- use registration
- with previously visited places (pose-nodes)
- to improve map quality

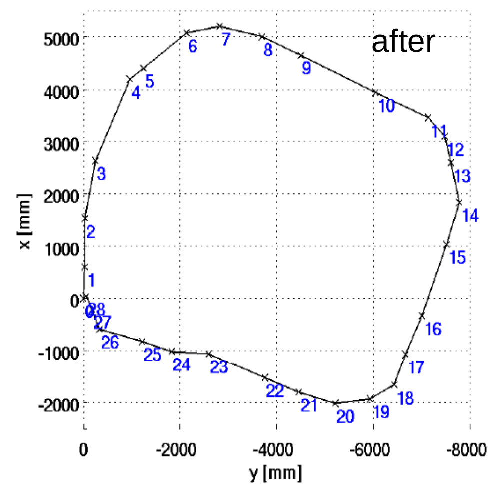












Cost Function

- edge
  - motion estimates (pose differences)
  - with uncertainty (note: each edge assumed to be independent)
  - imagine spring/damper of certain length & stiffness
- Mahalanobis Distance
  - N-dim distance between x and y
  - including uncertainty in form of covariance C
  - respectively, its inverse aka information matrix Ω

$$x = (x_1, x_2, \ldots, x_n)^T, \;\; y = (y_1, y_2, \ldots, y_n)^T$$

$$D_M(x, y) = \sqrt{(x - y)^T C^{-1} (x - y)}$$

Loop Closing & Relaxation

relaxation of the pose-graph

- find a configuration such that
  - the spring/damper network is
  - in an energetically optiomal configuration
- i.e., minimize (least squares sense)
- sum of Mahalanobis Distances

That‘s all, folks…