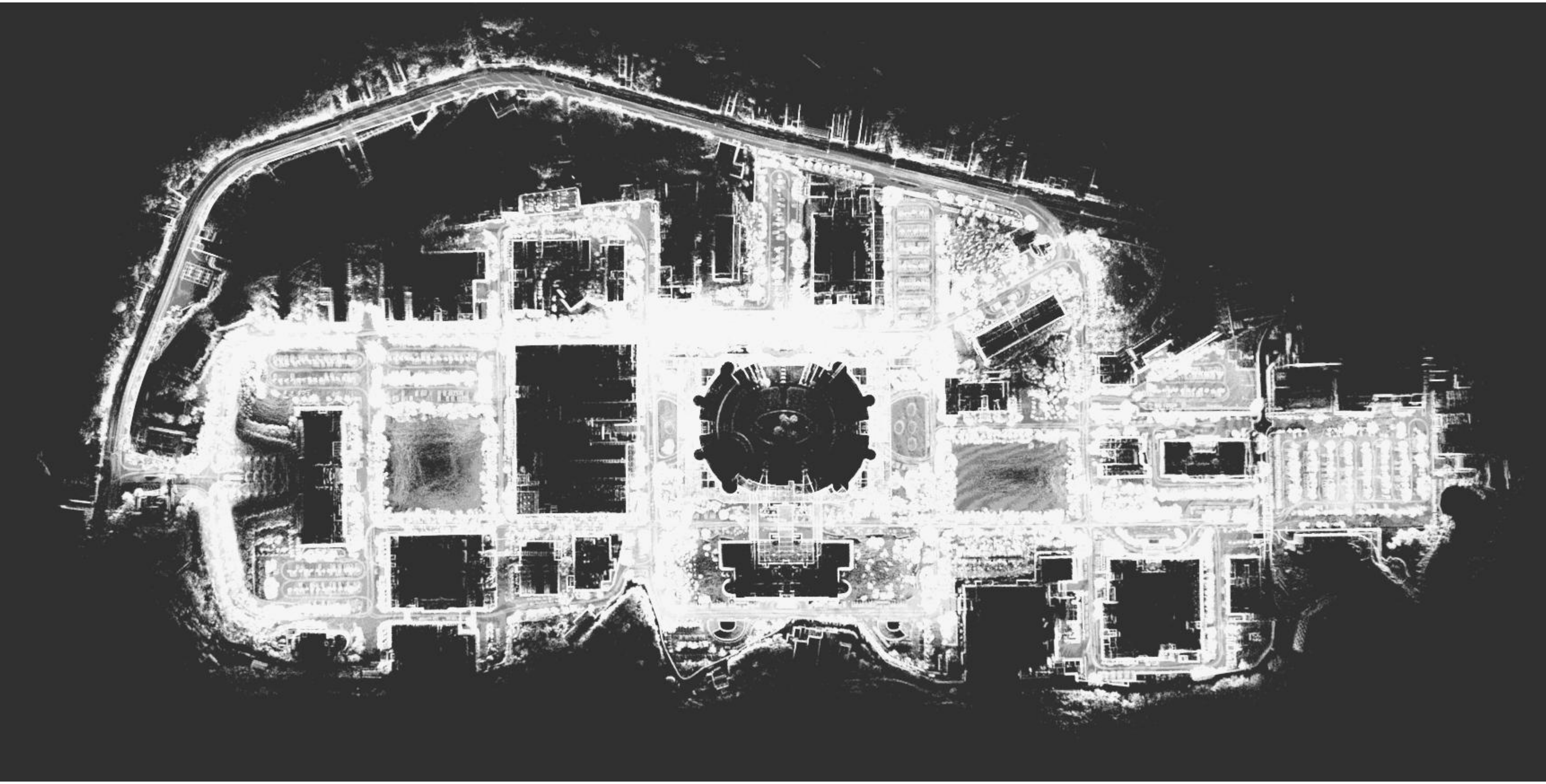
HCC Part 4 Lecture 2
Localization

Chieh-Chih (Bob) Wang 王傑智

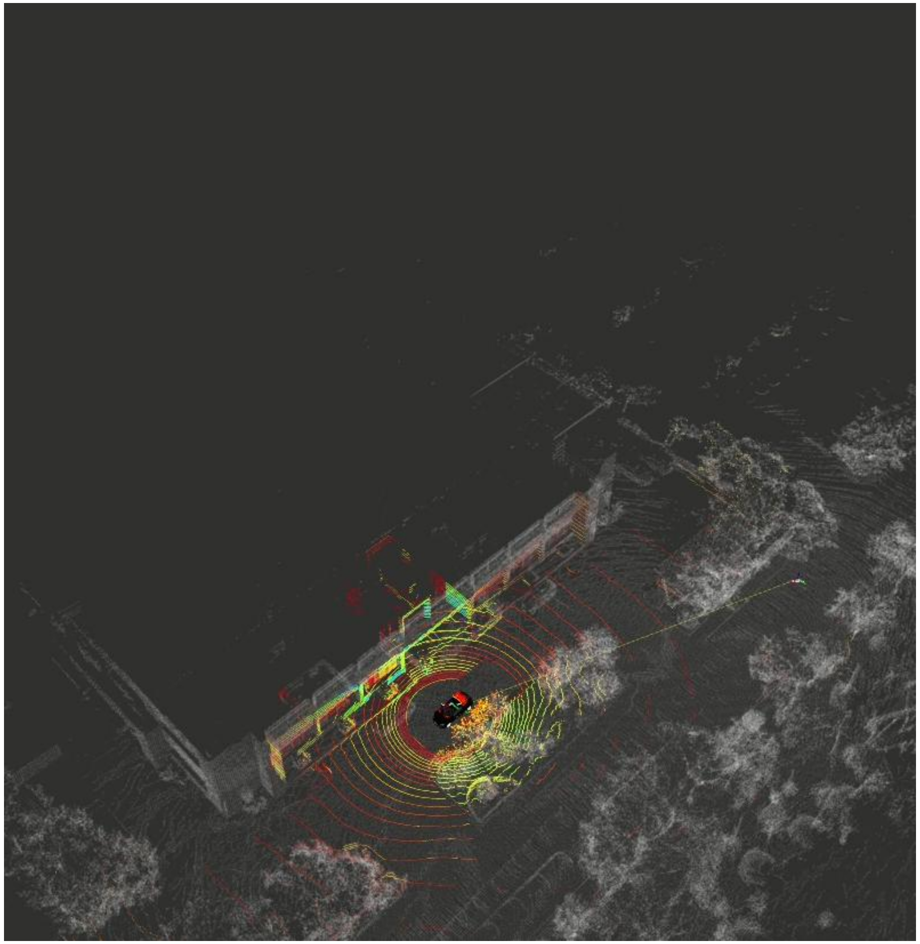
2025 Spring

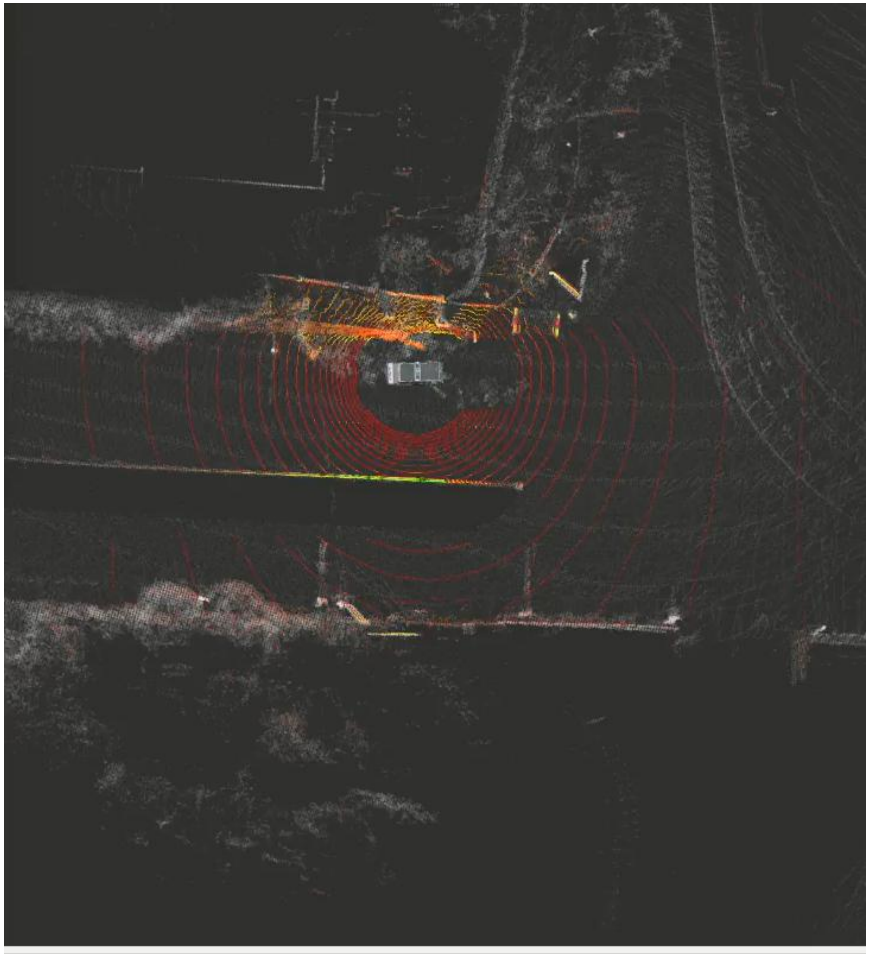
With slides by Roland Seigwart, & Illah R. Nourbakhsh

Geometric Mapping



Localization



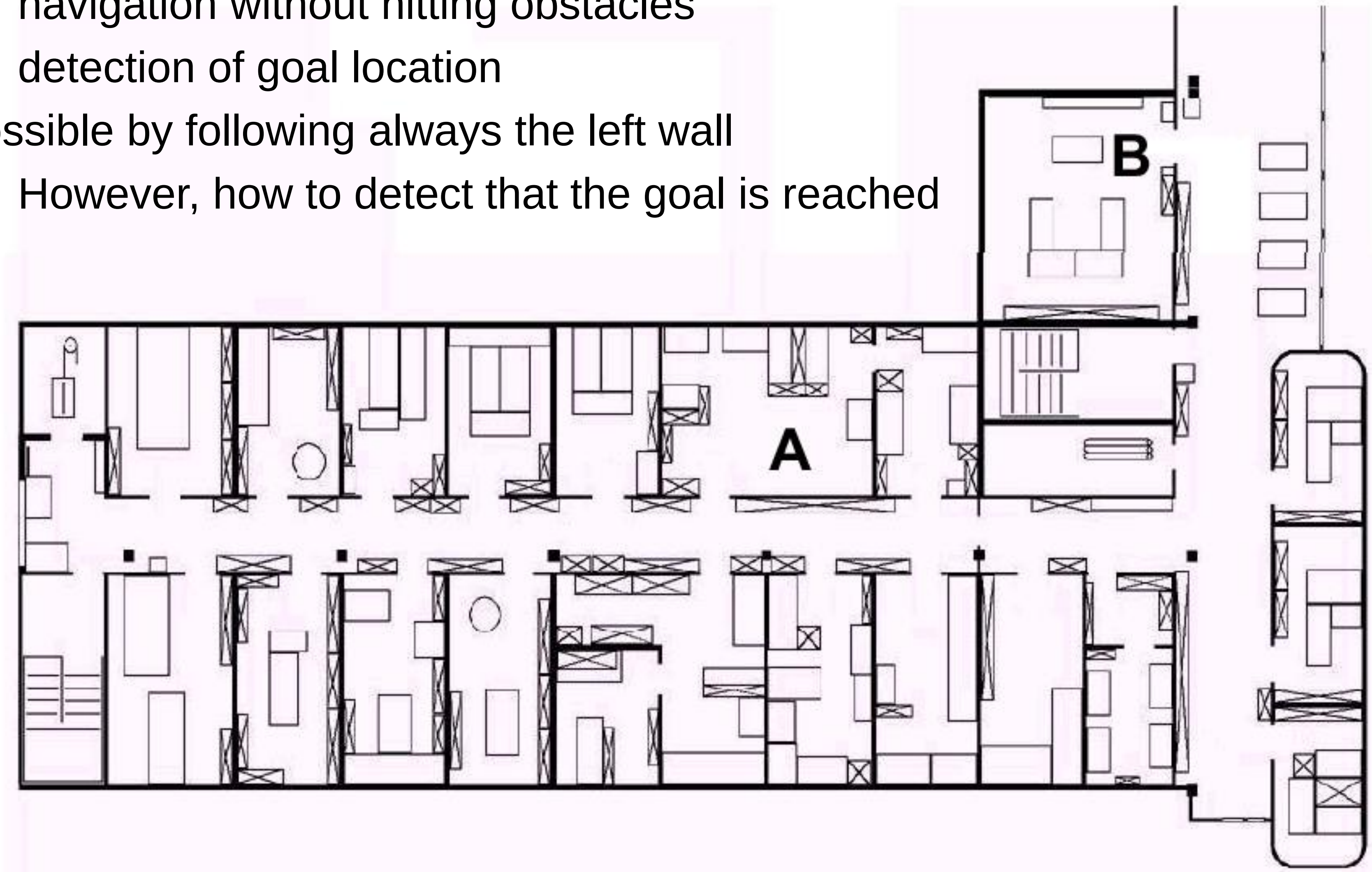


Today's Goals

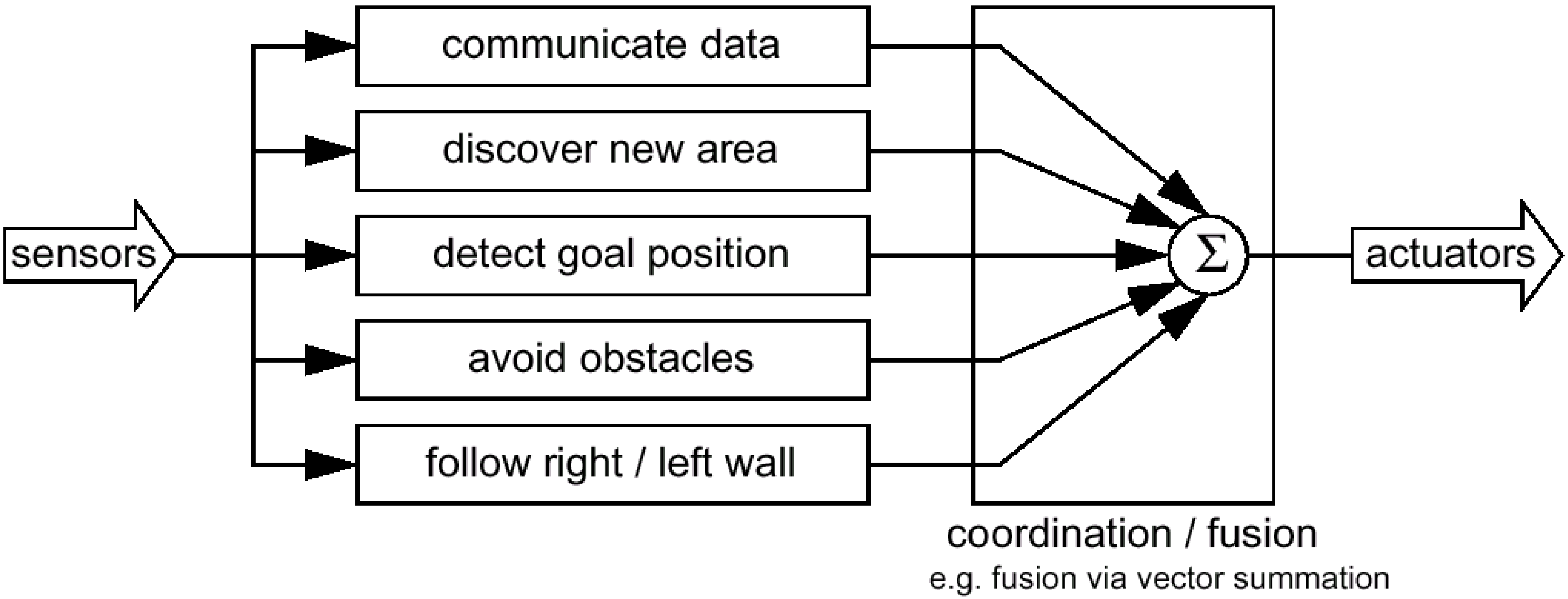
- Localization
- Belief & Map Representation
- Probabilistic Map-based Localization

To localize or not?

- How to navigate between A and B
  - navigation without hitting obstacles
  - detection of goal location
- Possible by following always the left wall
  - However, how to detect that the goal is reached

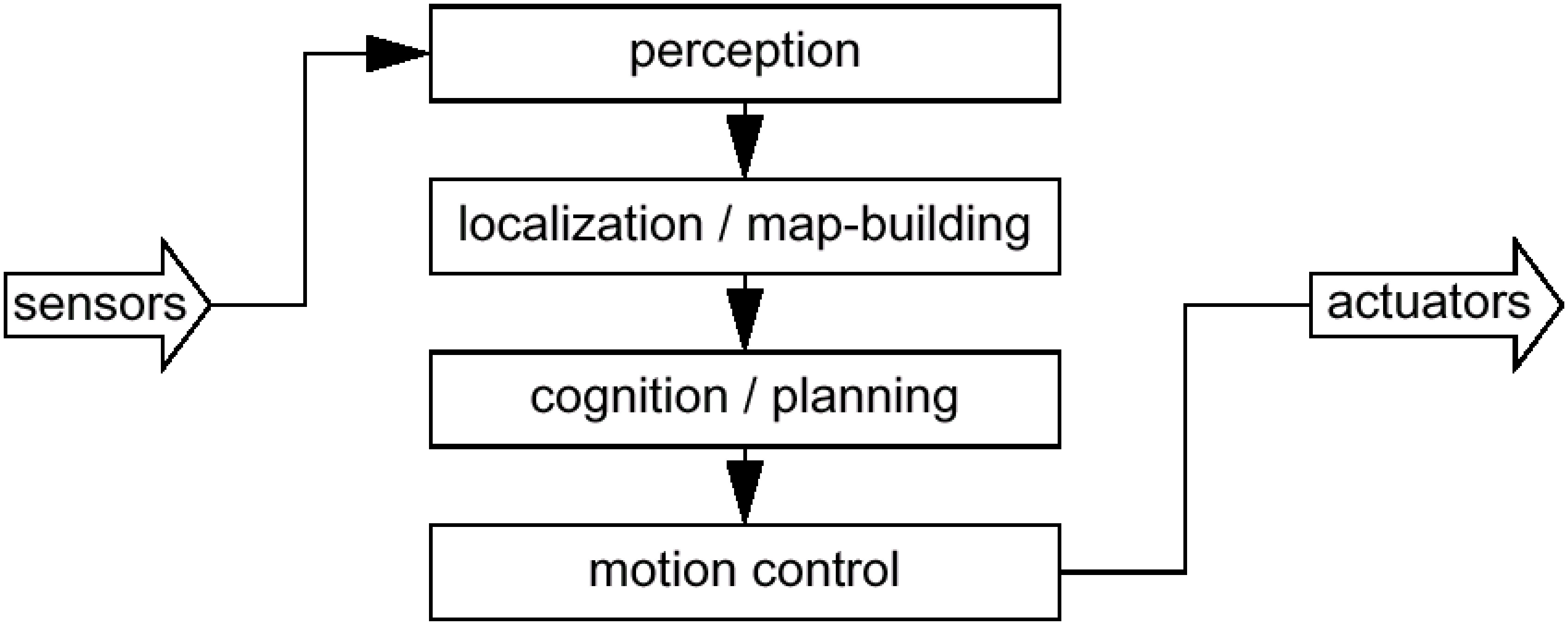


Behavior Based Navigation

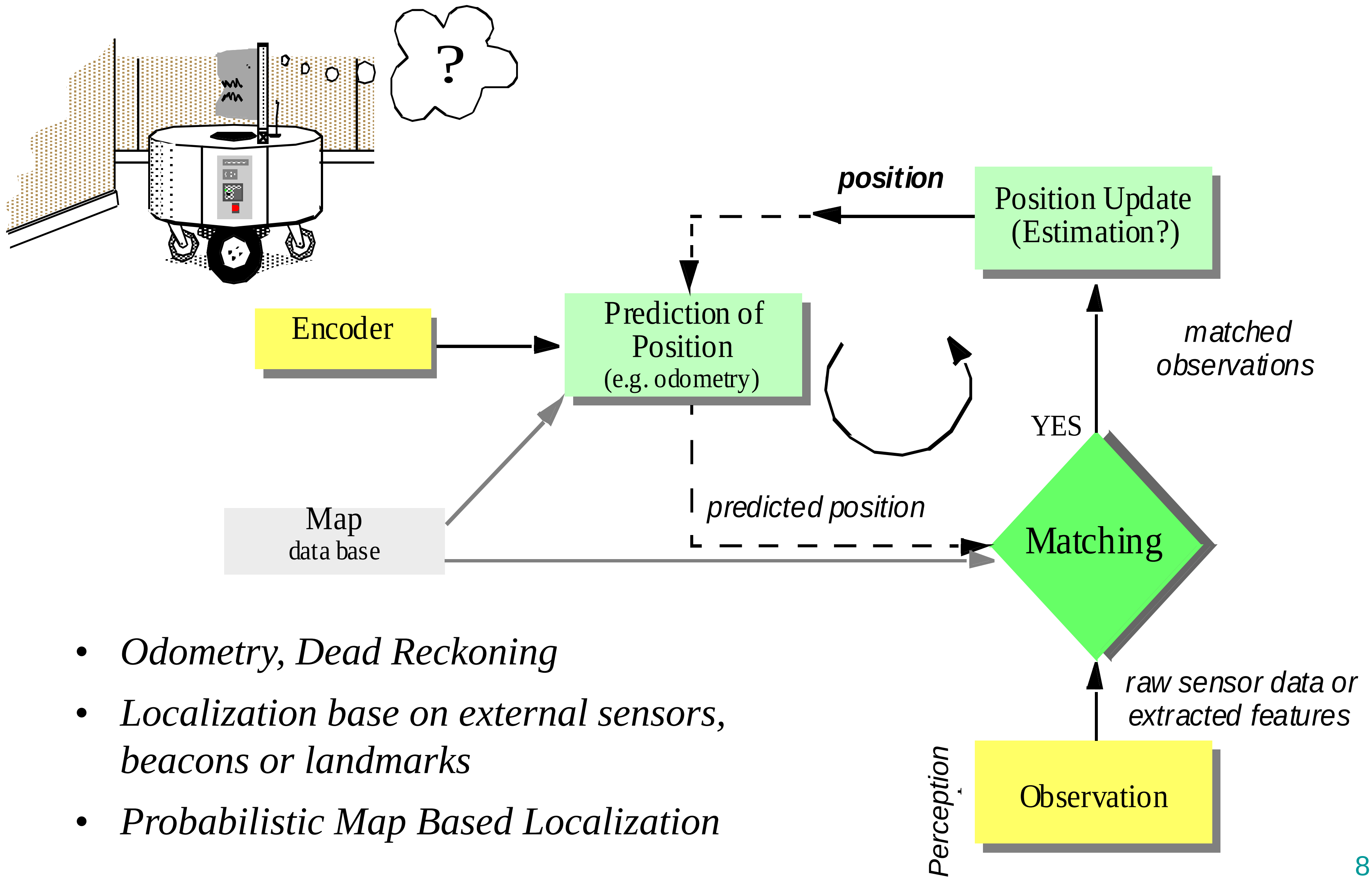


- Does not directly scale to other environments or to larger environments.
- The navigation code is location-specific
- must be carefully designed to produce the desired behaviors.
- system may have multiple active behaviors at any times.

Model Based Navigation



Localization, Where am I?



- *Odometry, Dead Reckoning*
- *Localization base on external sensors, beacons or landmarks*
- *Probabilistic Map Based Localization*

Challenges of Localization (Issues of GPS and other methods)

- Knowing the absolute position (e.g. GPS) is not sufficient
- Localization in human-scale in relation with environment
- Planning in the *Cognition* step requires more than only position as input
- Perception plays an important role

Odometry & Dead Reckoning

- Position update is based on proprioceptive sensors
  - Odometry: wheel sensors only
  - Dead reckoning: also heading sensors
- The movement of the robot, sensed with wheel encoders and/or heading sensors is integrated to the position.
  - Pros: Straight forward, easy
  - Cons: Errors are integrated -> unbound
- Using additional heading sensors (e.g. gyroscope) might help to reduce the cumulated errors, but the main problems remain the same.

Odometry: Error sources

deterministic (systematic) $\longleftrightarrow$ non-deterministic (non-systematic)

- deterministic errors can be eliminated by proper calibration of the system.
- non-deterministic errors have to be described by error models and will always leading to uncertain position estimate.

- Major Error Sources:
  - Limited resolution during integration (time increments, measurement resolution …)
  - Misalignment of the wheels (deterministic)
  - Unequal wheel diameter (deterministic)
  - Variation in the contact point of the wheel
  - Unequal floor contact (slipping, not planar …)
  - …

Classification of Integration Errors

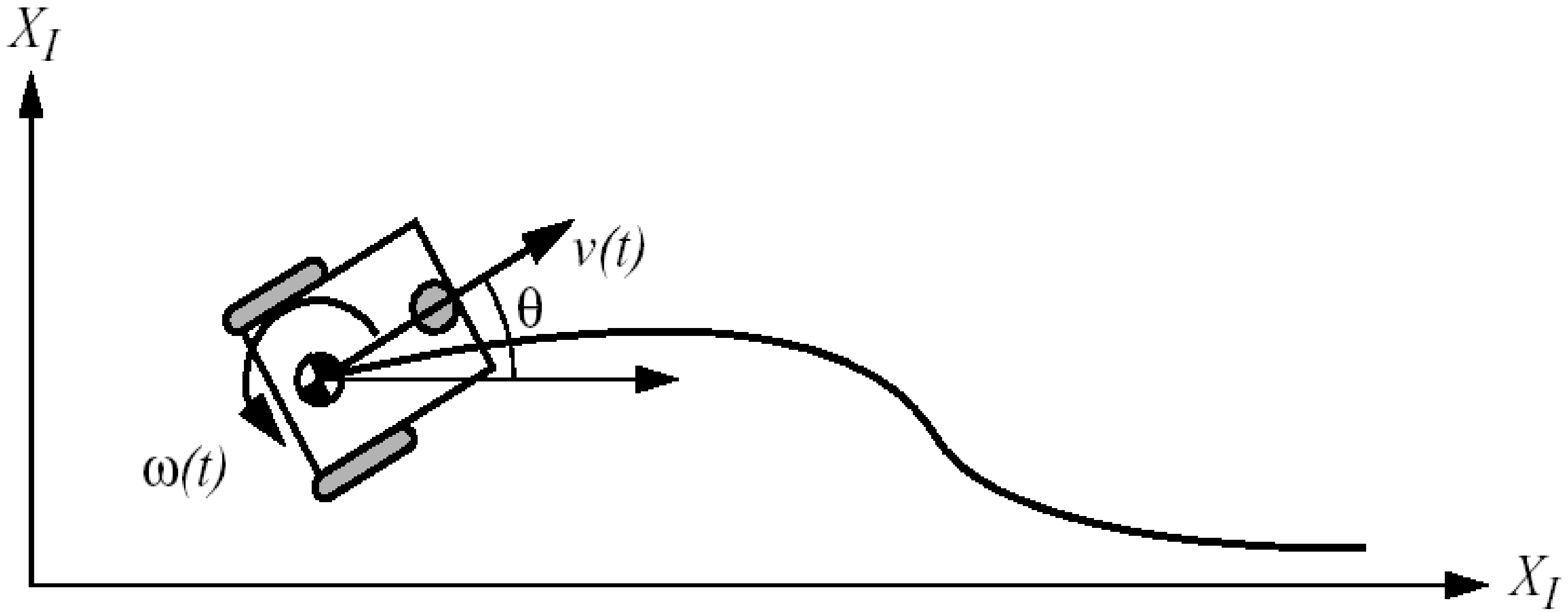
- Range error: integrated path length (distance) of the robots movement
  - sum of the wheel movements
- Turn error: similar to range error, but for turns
  - difference of the wheel motions
- Drift error: difference in the error of the wheels leads to an error in the robot's angular orientation

Over long periods of time, turn and drift errors far outweigh range errors!

  - Consider moving forward on a straight line along the **x** axis. The error in the **y**-position introduced by a move of **d** meters will have a component of $\mathbf{d}\boldsymbol{sin\Delta\theta}$, which can be quite large as the angular error $\Delta\theta$ grows.

Example: The Differential Drive Robot (1)

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$$



$(\Delta x;\Delta y;\Delta\theta)$ = path traveled in the last sampling interval;

$\Delta s_r;\Delta s_l$ = traveled distances for the right and left wheel respectively;

b = distance between the two wheels of differential-drive robot.

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \qquad \Delta x = \Delta s \cos(\theta + \Delta\theta/2)$$

$$\Delta y = \Delta s \sin(\theta + \Delta\theta/2)$$

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2}$$

$$p' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = p + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta/2) \\ \Delta s \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta/2) \\ \Delta s \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix}$$

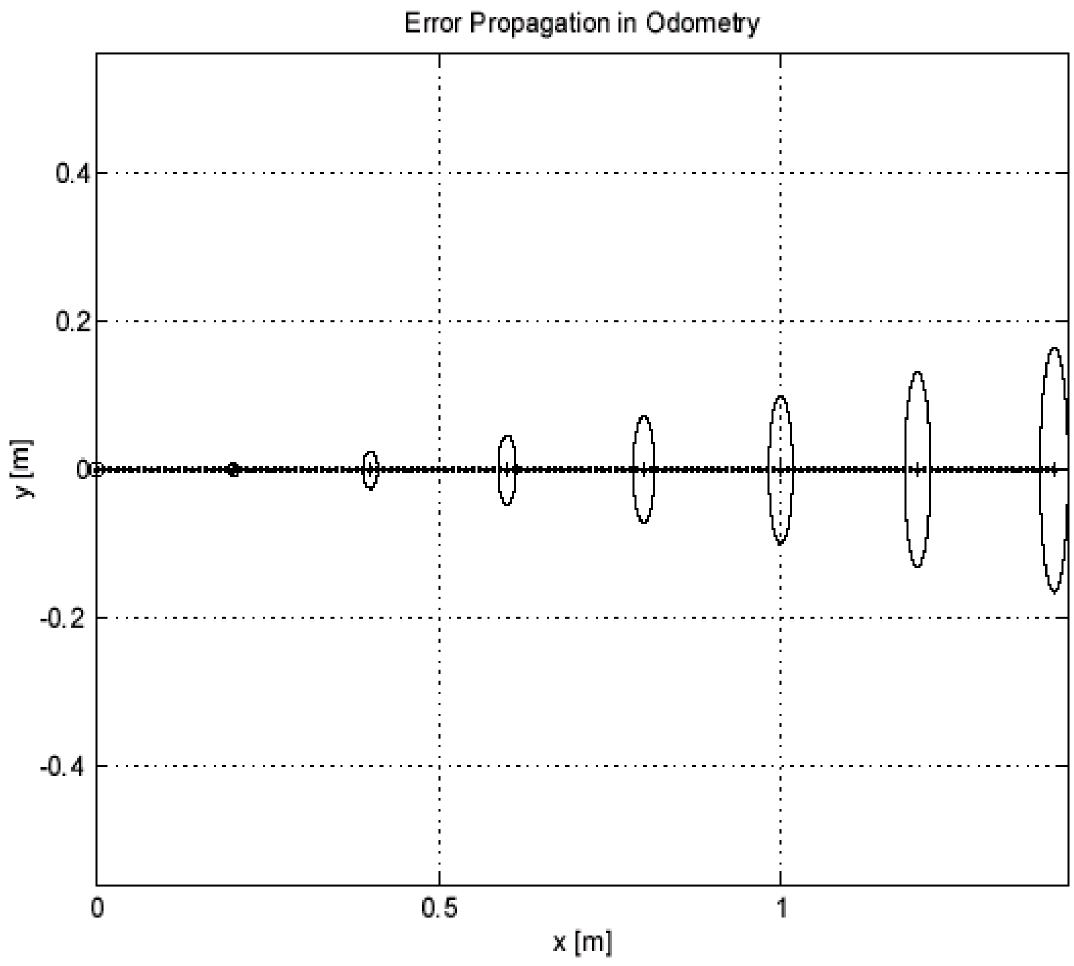
$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

$$\Sigma_\Delta = covar(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$$

Odometry

Growth of Pose uncertainty for Straight Line Movement

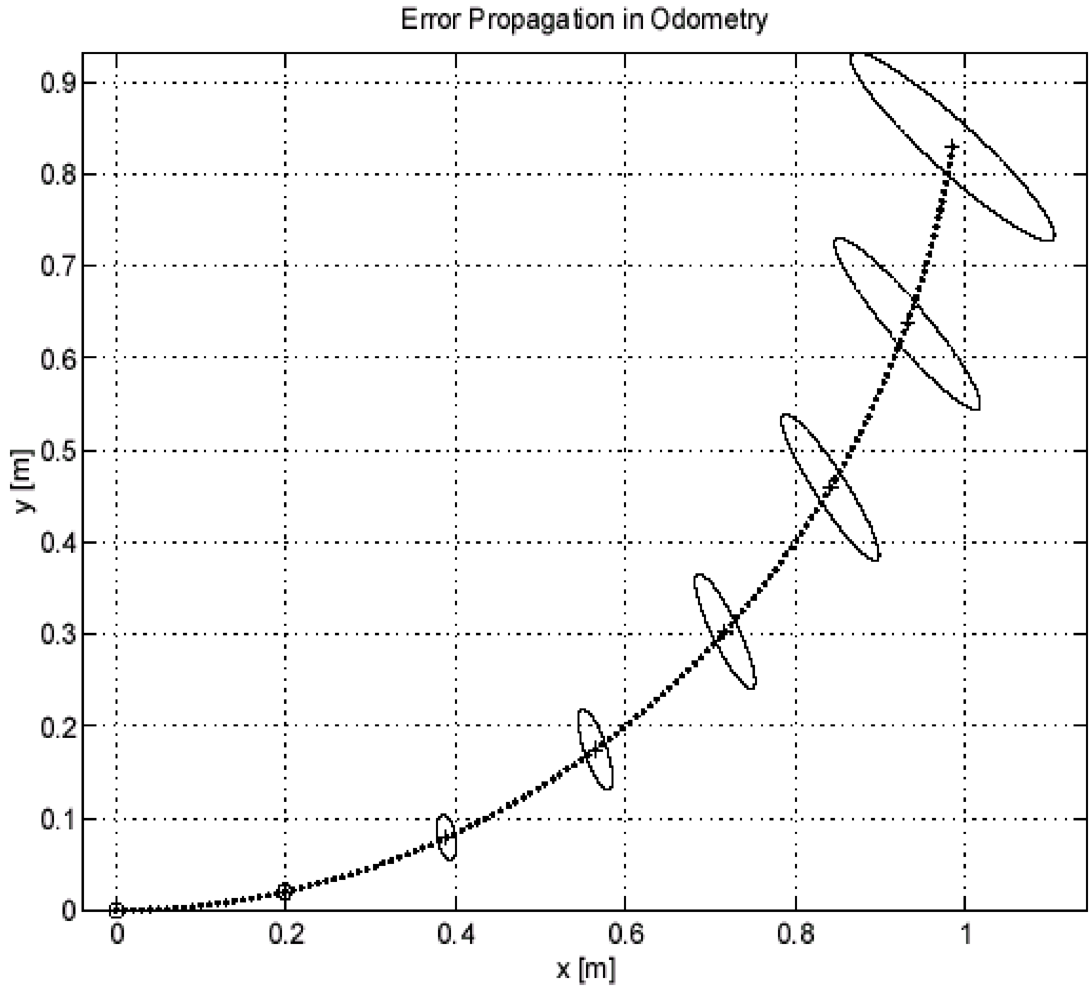
- Note: Errors perpendicular to the direction of movement are growing much faster!



Odometry

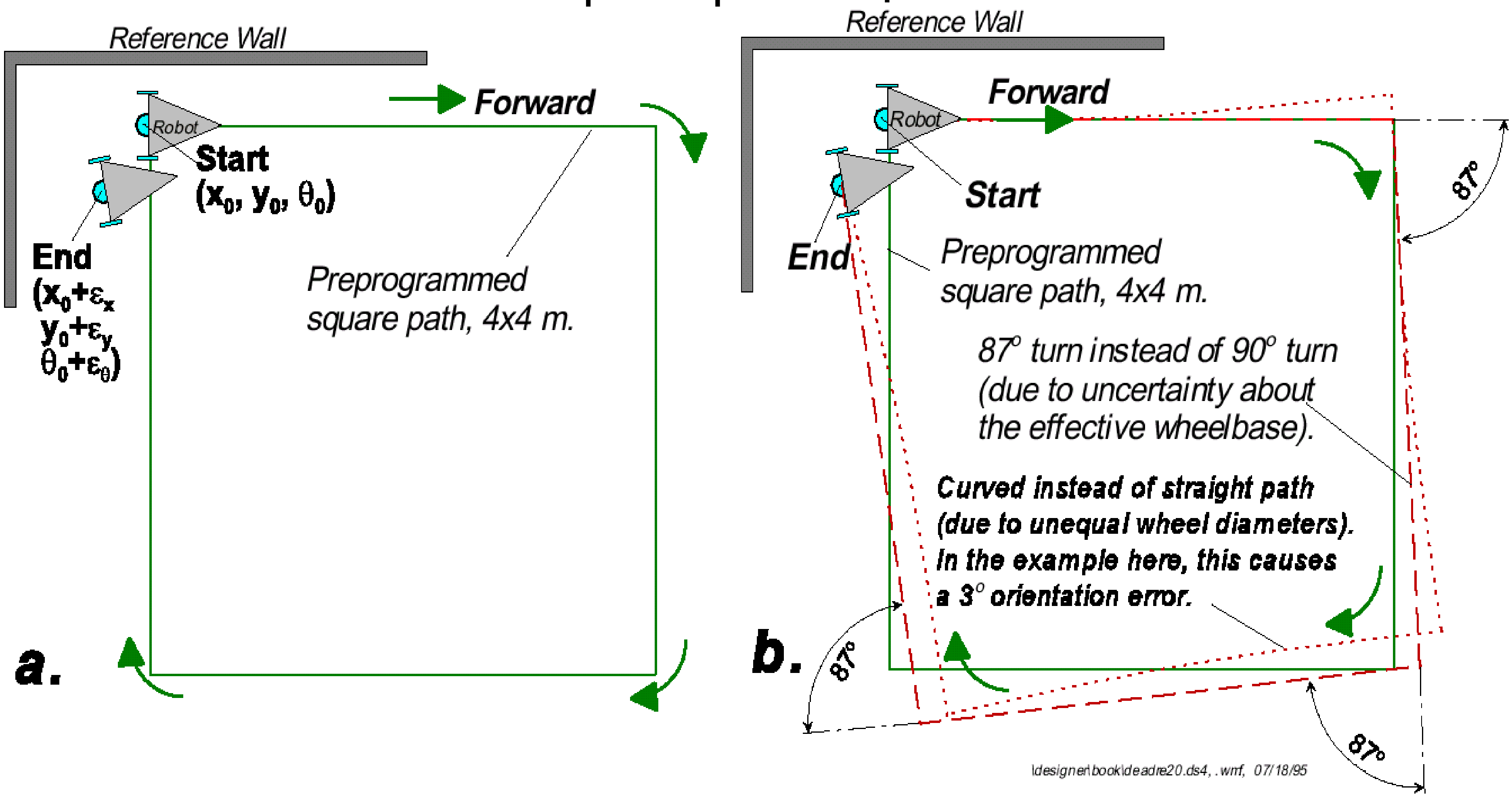
Growth of Pose uncertainty for Movement on a Circle

- Note: Errors ellipse does not remain perpendicular to the direction of movement!



Odometry: Calibration of Errors I

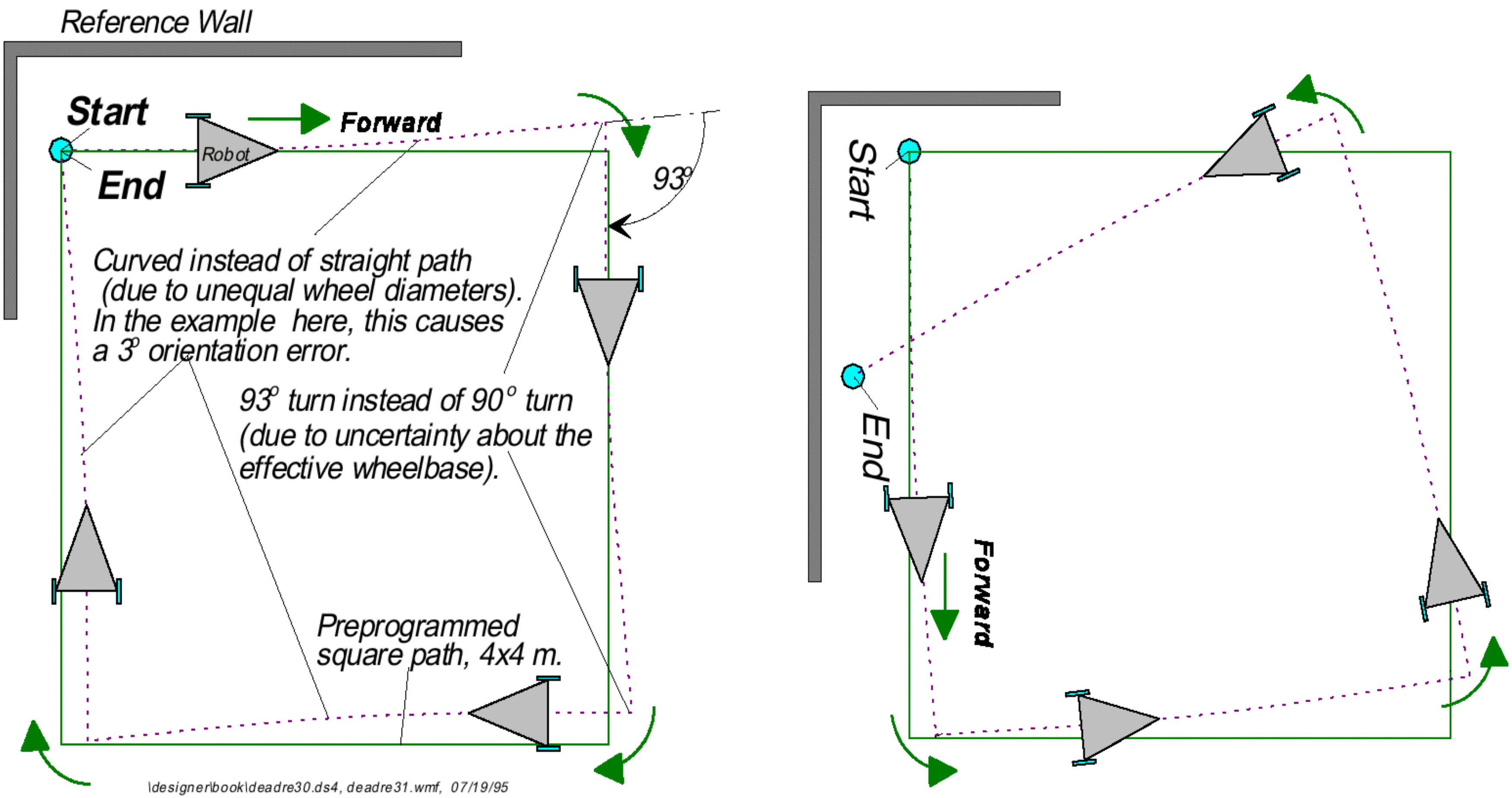
- The unidirectional square path experiment



Ref page 130-150 : J. Borenstein and L. Feng. Umbmark - a method for measuring, comparing, and correcting dead-reckoning errors in mobile robots. Technical Report UM-MEAM-94-22, University of Michigan, 1994.

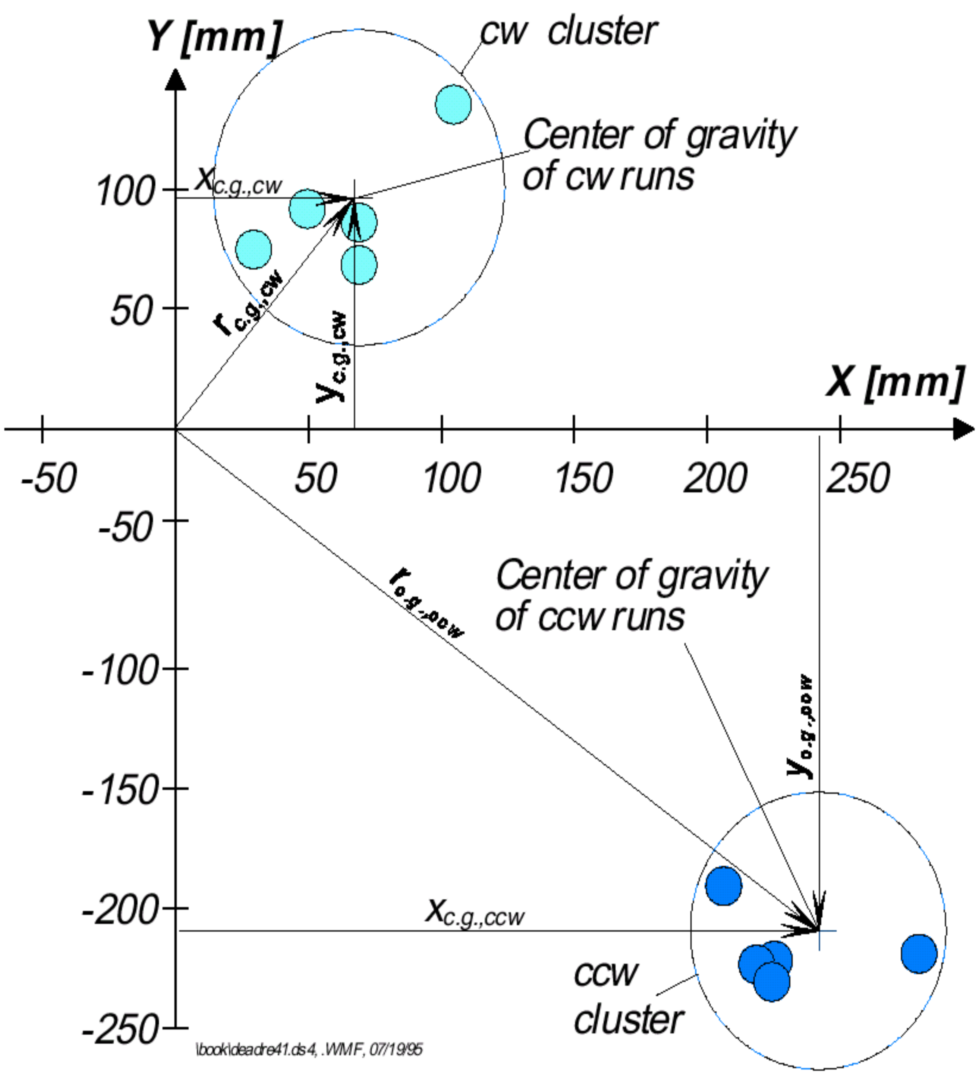
Odometry: Calibration of Errors II

- The bi-directional square path experiment



Odometry: Calibration of Errors III

- The deterministic and non-deterministic errors



- Alonzo Kelly, "Linearized Error Propagation in Odometry", International Journal of Robotics Research, Vol 23, No 2, Feb 2004, pp 179-218.

- Alonzo Kelly, “Fast and Easy Systematic and Stochastic Odometry Calibration”. IROS, Sendai Japan, September 2004.

Today's Goals

- Localization
- Belief & Map Representation
- Probabilistic Map-based Localization
- Other Localization Methods (not probabilistic)

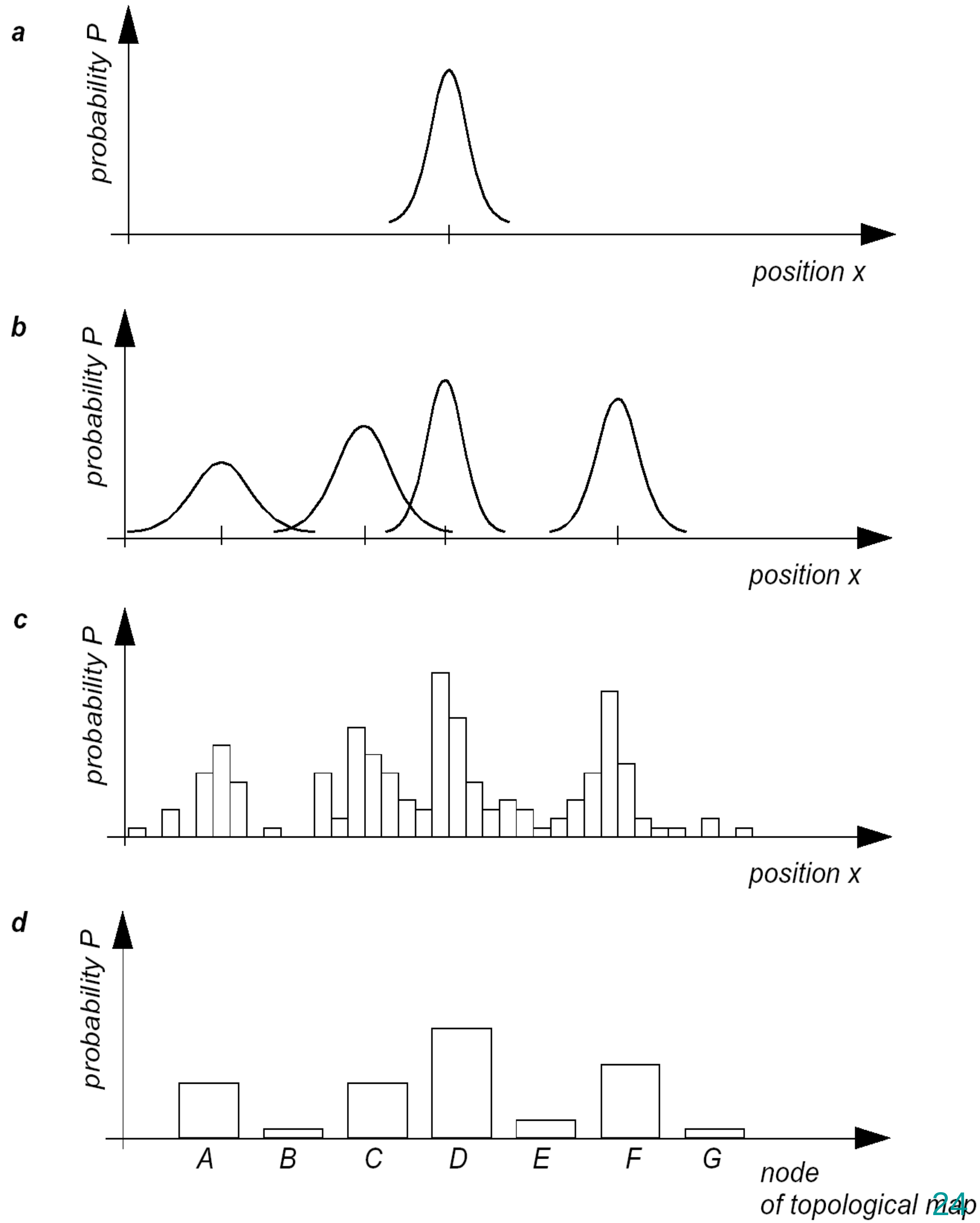
Belief & Map Representation

- *Map Representation*: the robot must have a representation (model) of the environment.
  - Map-based localization: the issue of representation
- *Belief Representation:* the robot must also have a representation of its belief regarding its position on the map.

- Decisions along these two design axes can result in varying levels of architectural complexity, computational complexity, and overall localization accuracy.

Belief Representation

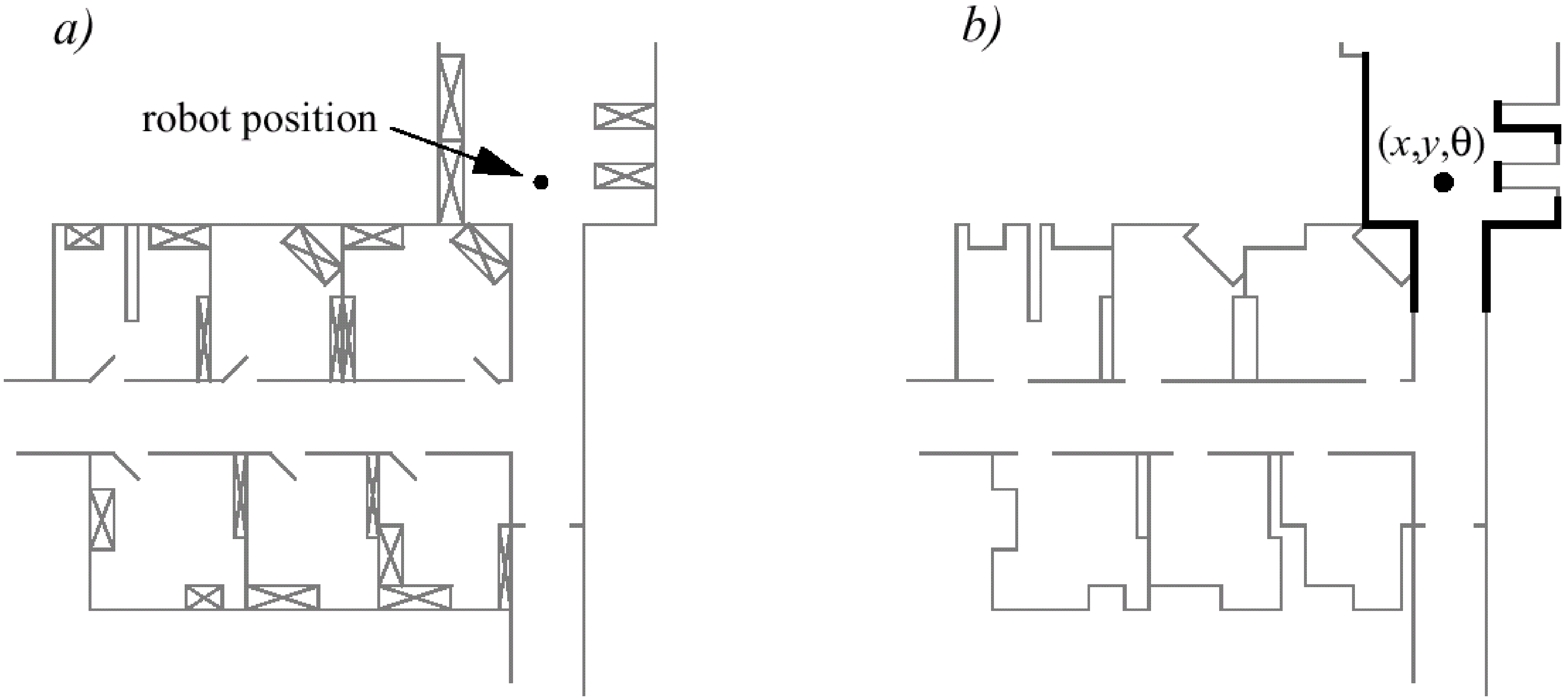
- a) Continuous map with *single hypothesis*
- b) Continuous map with *multiple hypothesis*
- d) Discretized map with probability distribution
- d) Discretized topological map with probability distribution



Belief Representation: Characteristics

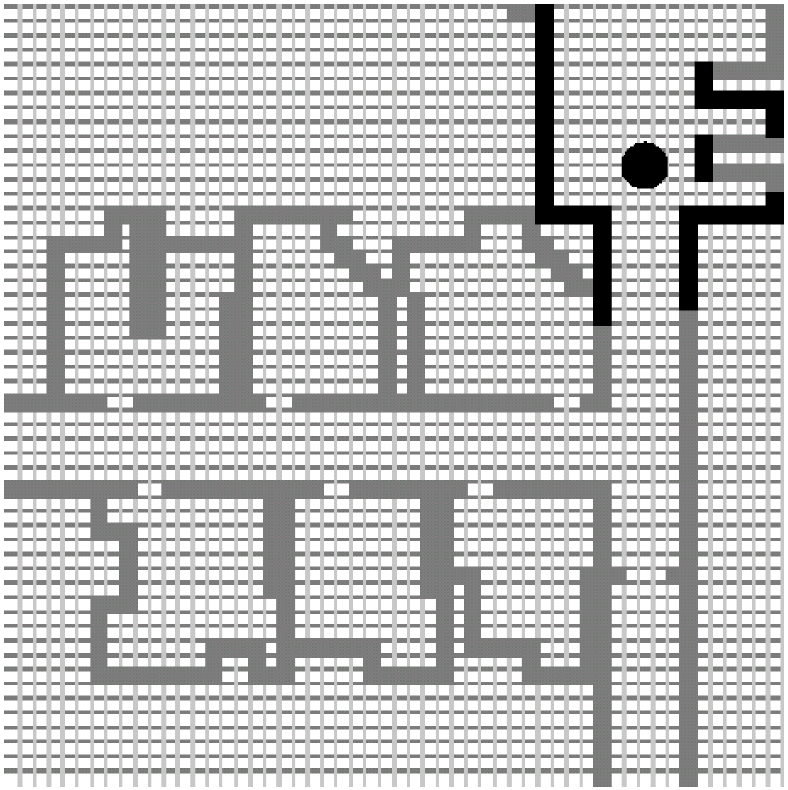
- Continuous
  - Precision bound by sensor data
  - Typically single hypothesis pose estimate
  - Lost when diverging (for single hypothesis)
  - Compact representation and typically reasonable in processing power.
- Discrete
  - Precision bound by resolution of discretisation
  - Typically multiple hypothesis pose estimate
  - Never lost (when diverges converges to another cell)
  - Important memory and processing power needed. (not the case for topological maps)

Single-hypothesis Belief – **Continuous Line-Map**

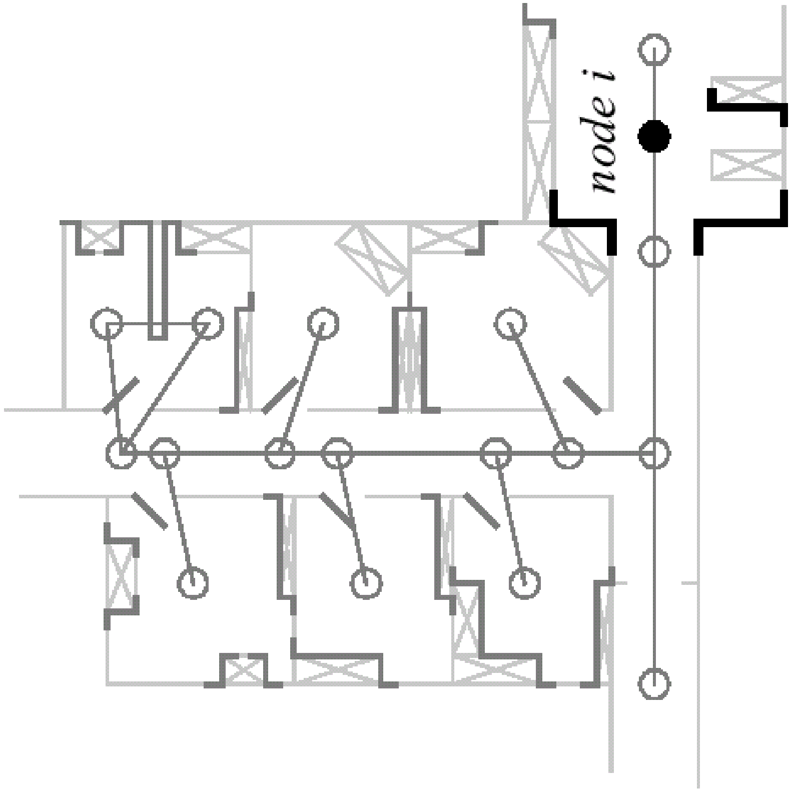


Single-hypothesis Belief – **Grid and Topological Map**

c)



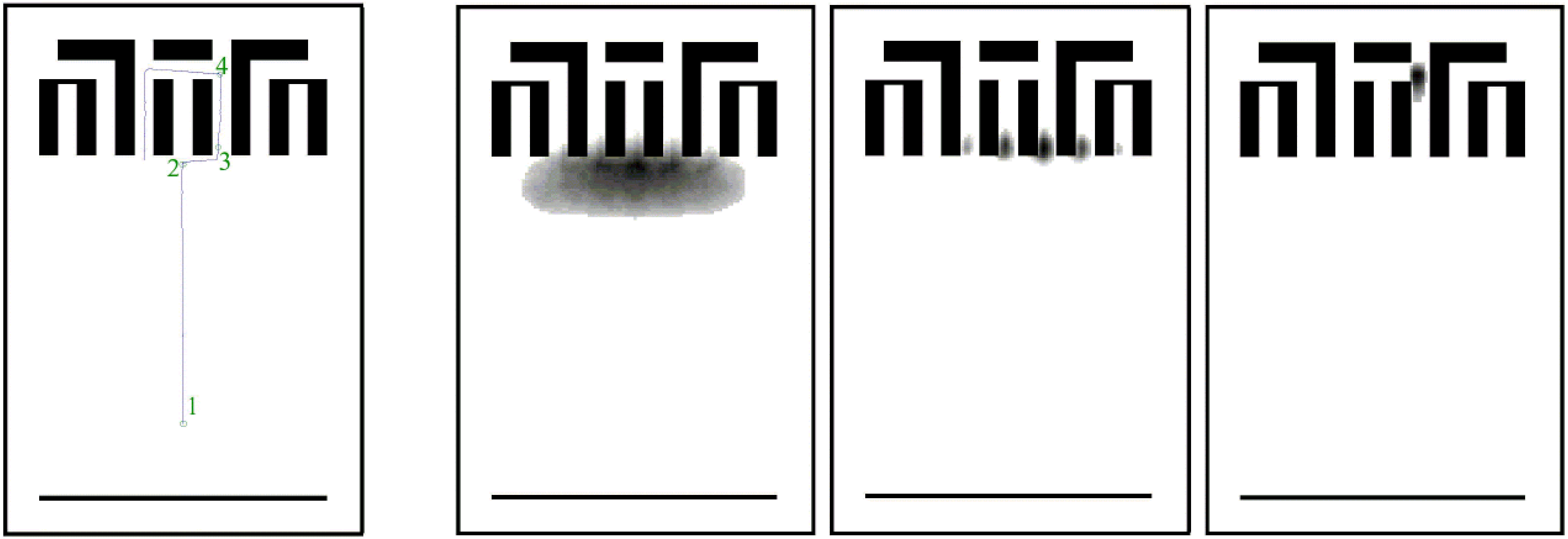
d)



Grid-based Representation – Multi Hypotheses

- Grid size around 20 cm^2.

Courtesy of W. Burgard



Path of the robot

Belief states at positions 2, 3 and 4

Map Representation

1. Map precision vs. application

2. Features precision vs. map precision

3. Precision vs. computational complexity

- Continuous Representation
- Decomposition (Discretization)

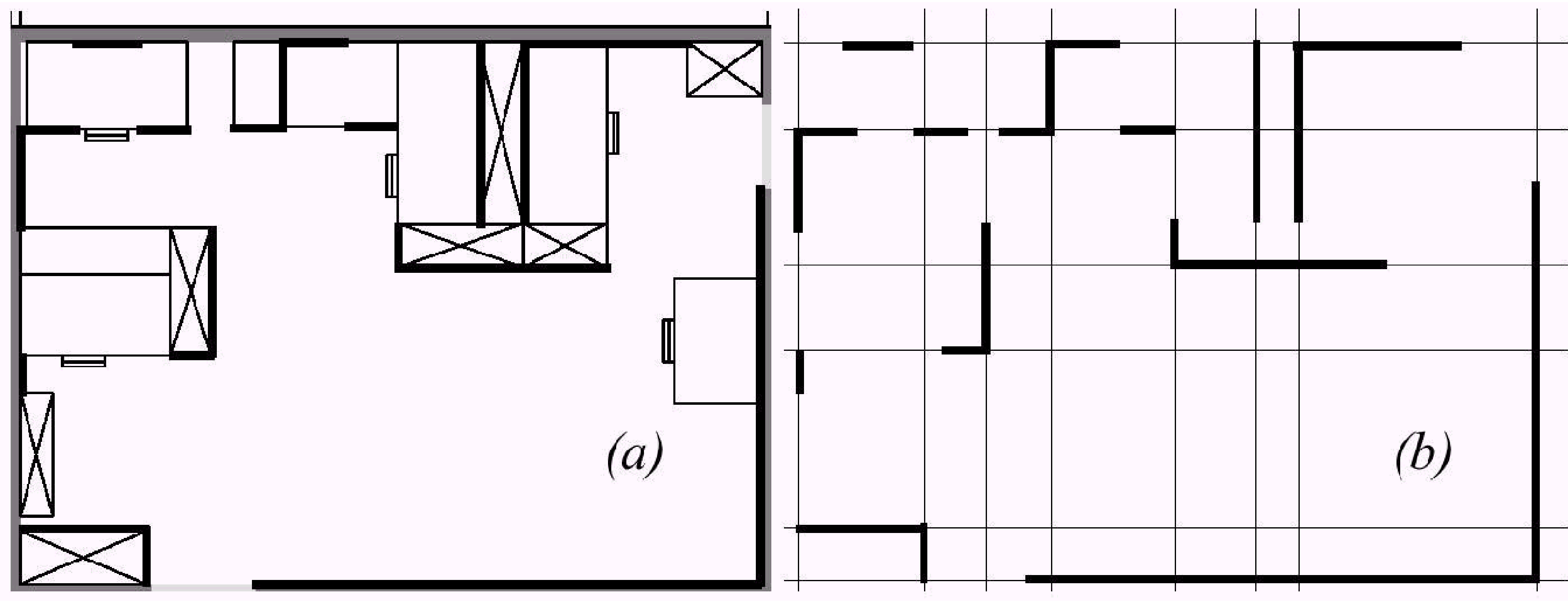
Representation of the Environment

- Environment Representation
  - Continuos Metric $\rightarrow$ x,y,θ
  - Discrete Metric $\rightarrow$ metric grid
  - Discrete Topological $\rightarrow$ topological grid
- Environment Modeling
  - **Raw sensor data**, e.g. laser range data, grayscale images
    - large volume of data, low distinctiveness on the level of individual values
    - makes use of all acquired information
  - **Low level features**, e.g. line other geometric features
    - medium volume of data, average distinctiveness
    - filters out the useful information, still ambiguities
  - **High level features**, e.g. doors, a car, the Eiffel tower
    - low volume of data, high distinctiveness
    - filters out the useful information, few/no ambiguities, not enough information

Map Representation: Continuous Line-Based

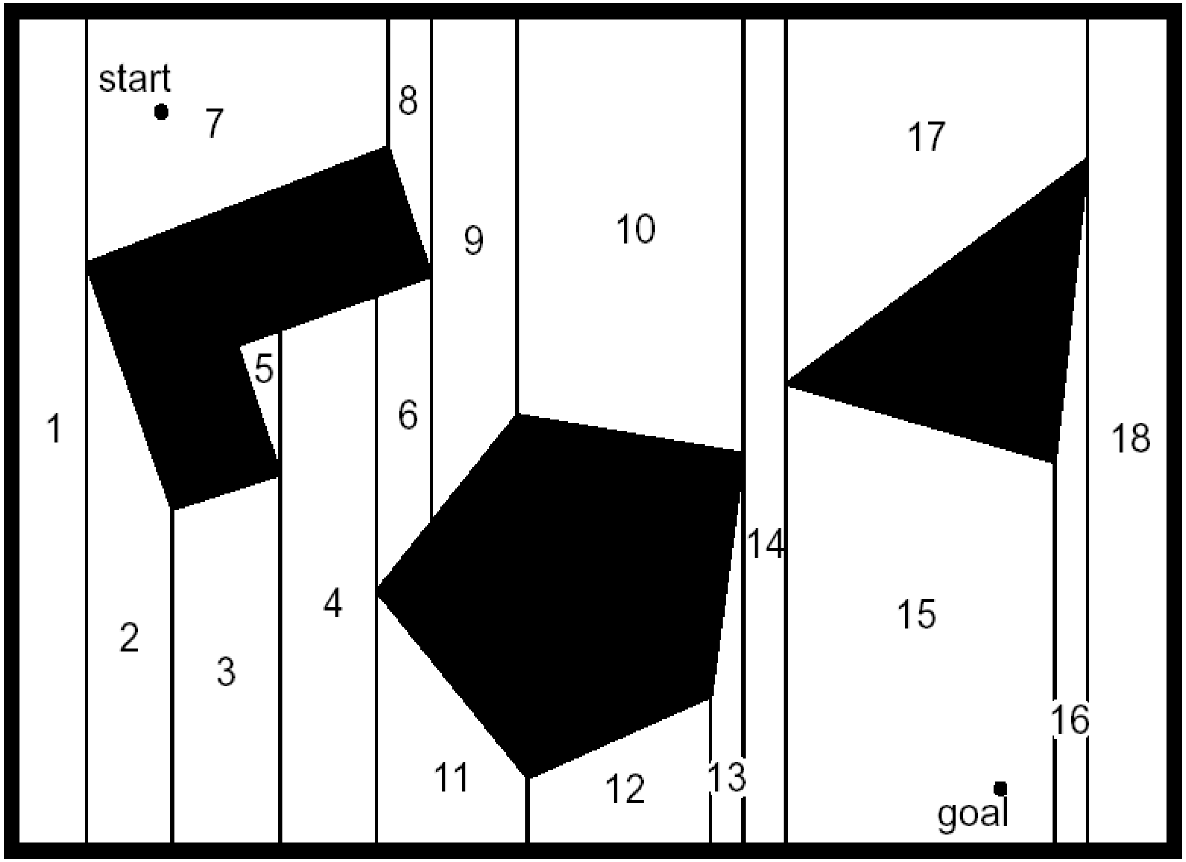
a) Architecture map

b) Representation with set of infinite lines



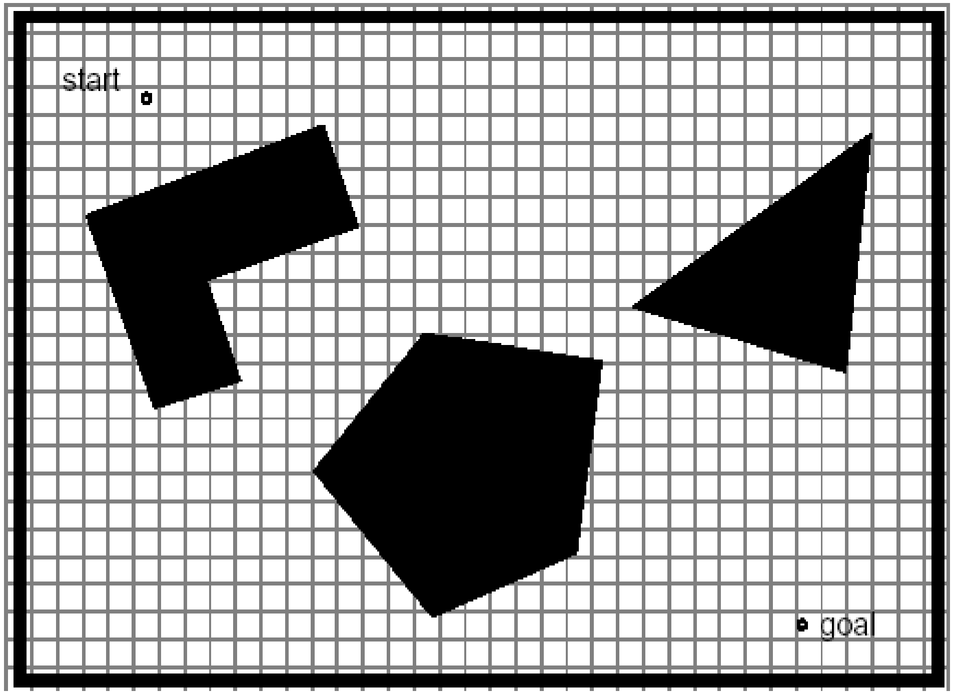
Map Representation: Decomposition (1)

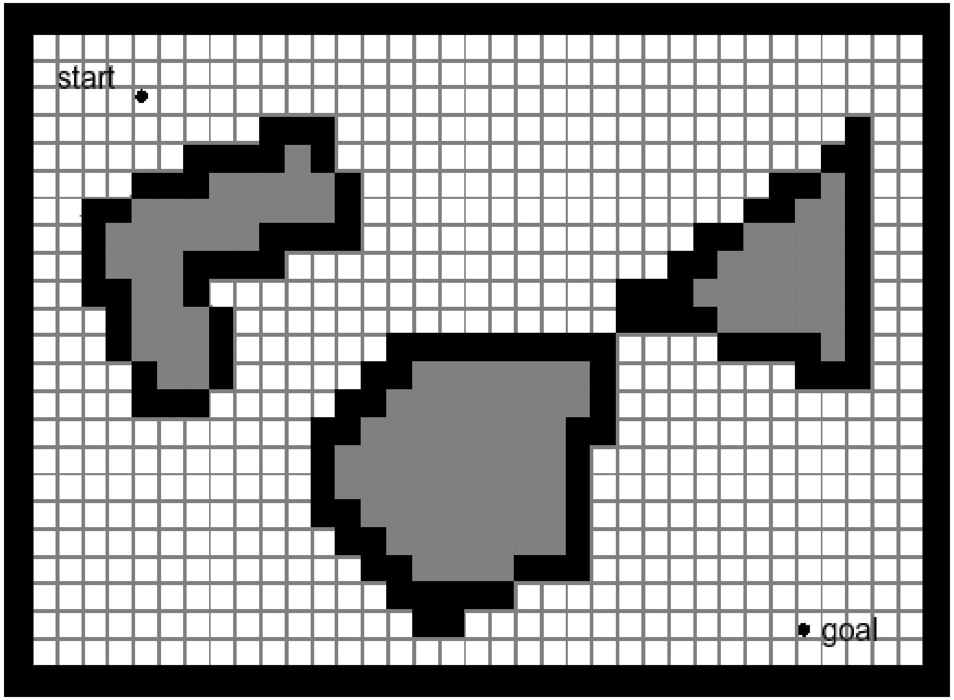
- Exact cell decomposition



Map Representation: Decomposition (2)

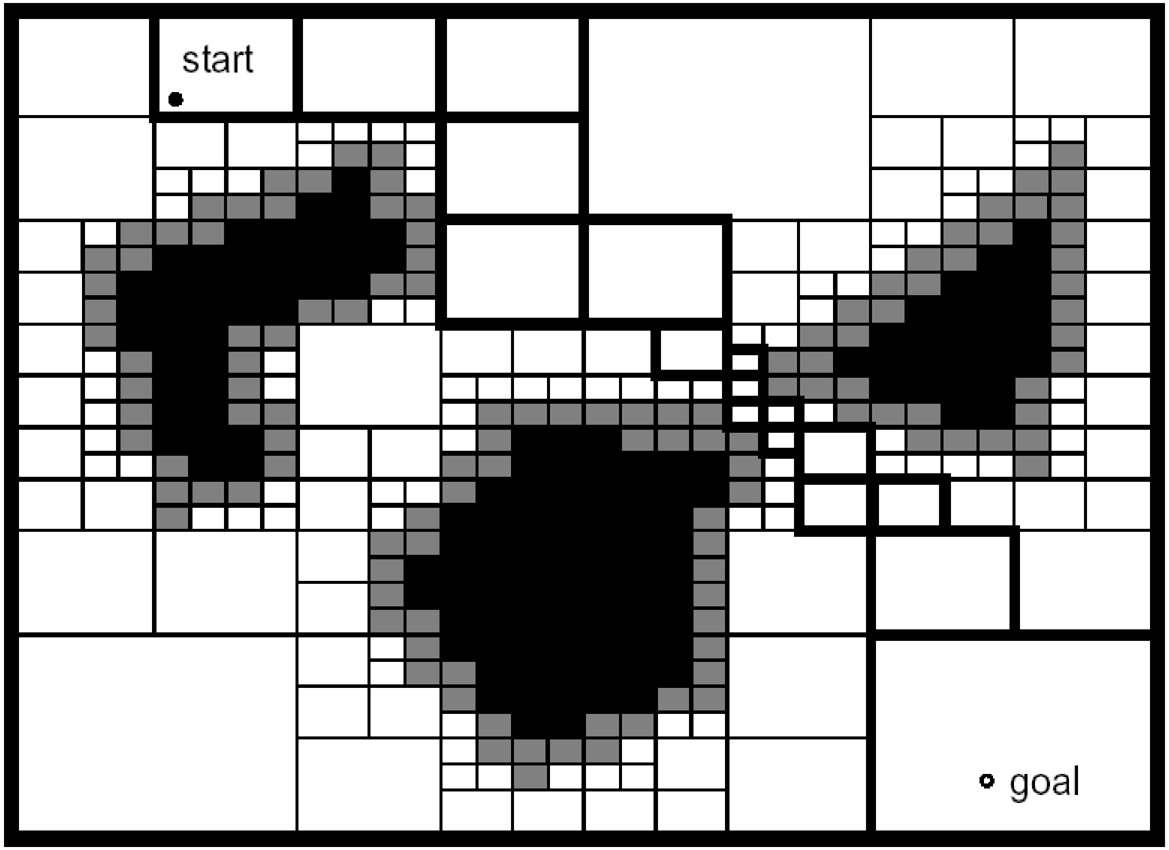
- Fixed cell decomposition





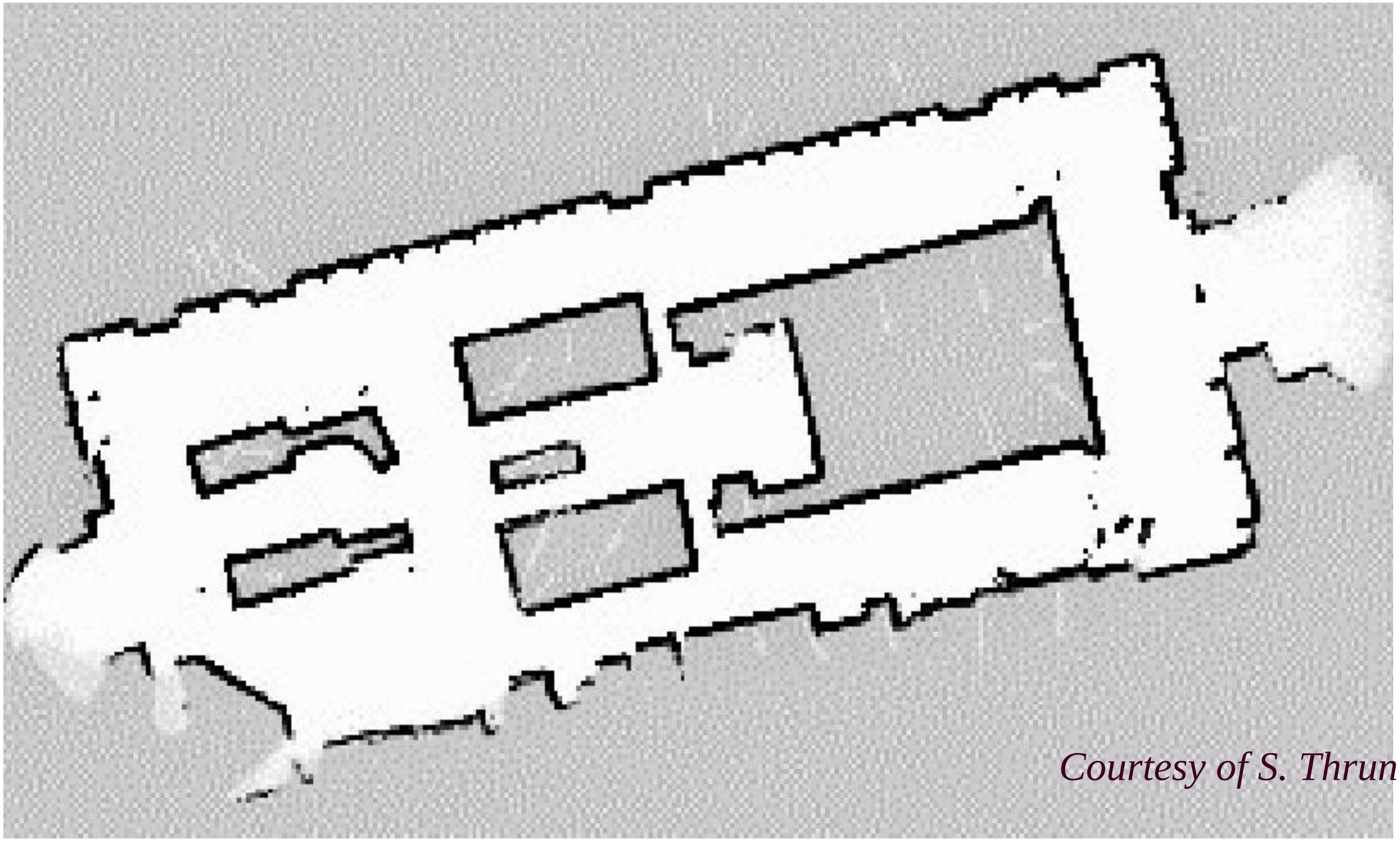
Map Representation: Decomposition (3)

- Adaptive cell decomposition



Map Representation: Decomposition (4)

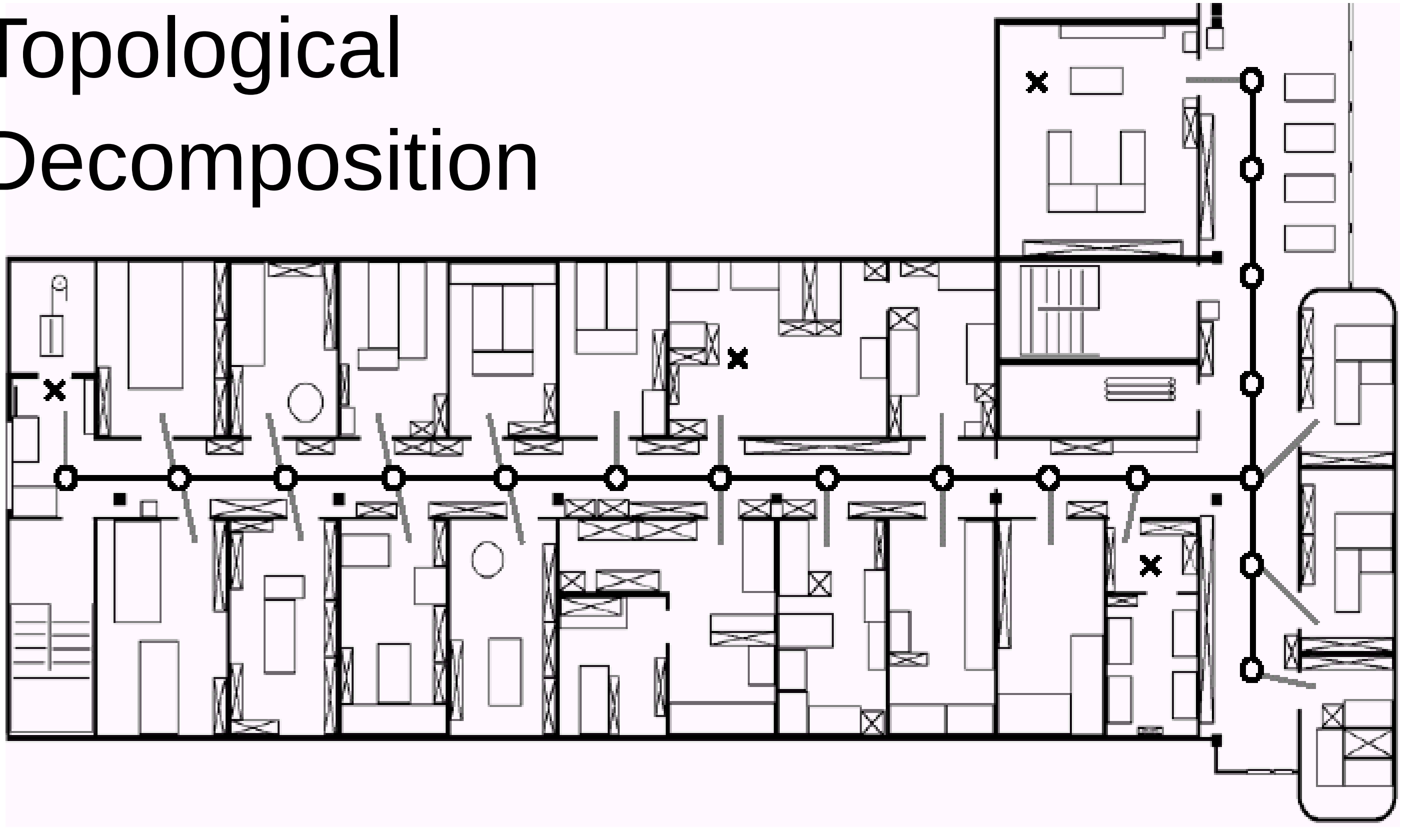
- Fixed cell decomposition – Example with very small cells



Courtesy of S. Thrun

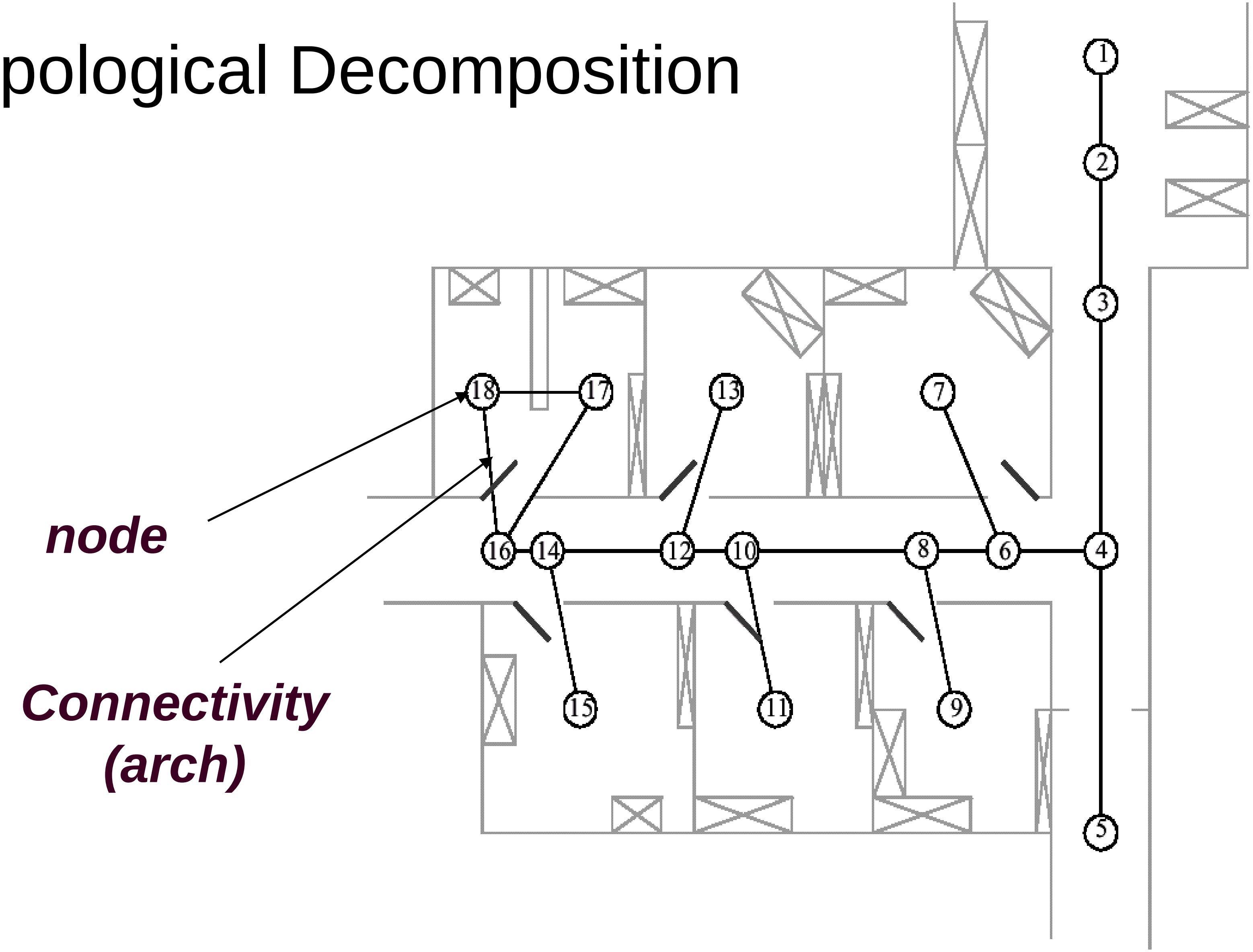
Map Representation: Decomposition (5)

- Topological Decomposition



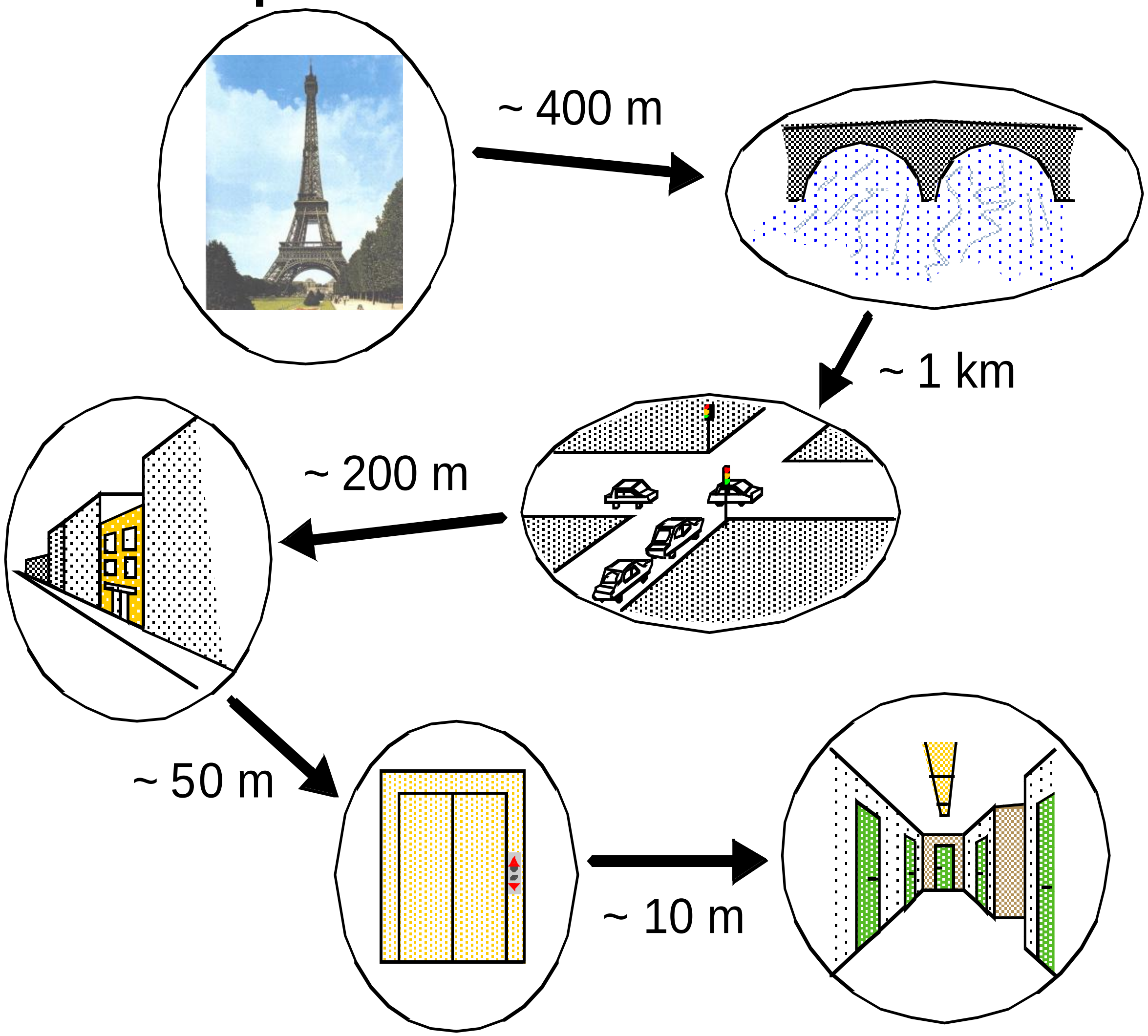
Map Representation: Decomposition (6)

- Topological Decomposition



Map Representation: Decomposition (7)

- Topological Decomposition



~~State-of-the-Art:~~

~~Current~~ Challenges (?) in Map Representation

- Real world is dynamic
- Perception is still a major challenge
  - Error prone
  - Extraction of useful information difficult
- Traversal of open space
- How to build up topology (boundaries of nodes)
- Sensor fusion

Today’s Goals

- Localization
- Belief & Map Representation
- Probabilistic Map-based Localization

Probabilistic, Map-Based Localization

- Consider a mobile robot moving in a known environment.
- As it start to move, say from a precisely known location, it might keep track of its location using odometry.
- However, after a certain movement the robot will get very uncertain about its position.

→ update using an observation of its environment.

- observation lead also to an estimate of the robots position which can than be fused with the odometric estimation to get the best possible update of the robots actual position.

Probabilistic, Map-Based Localization

- Action update
  - action model ACT

$$s'_t = Act(o_t, s_{t-1})$$

  with $\mathbf{o_t}$: Encoder Measurement, $\mathbf{s_{t-1}}$: prior belief state

  - increases uncertainty

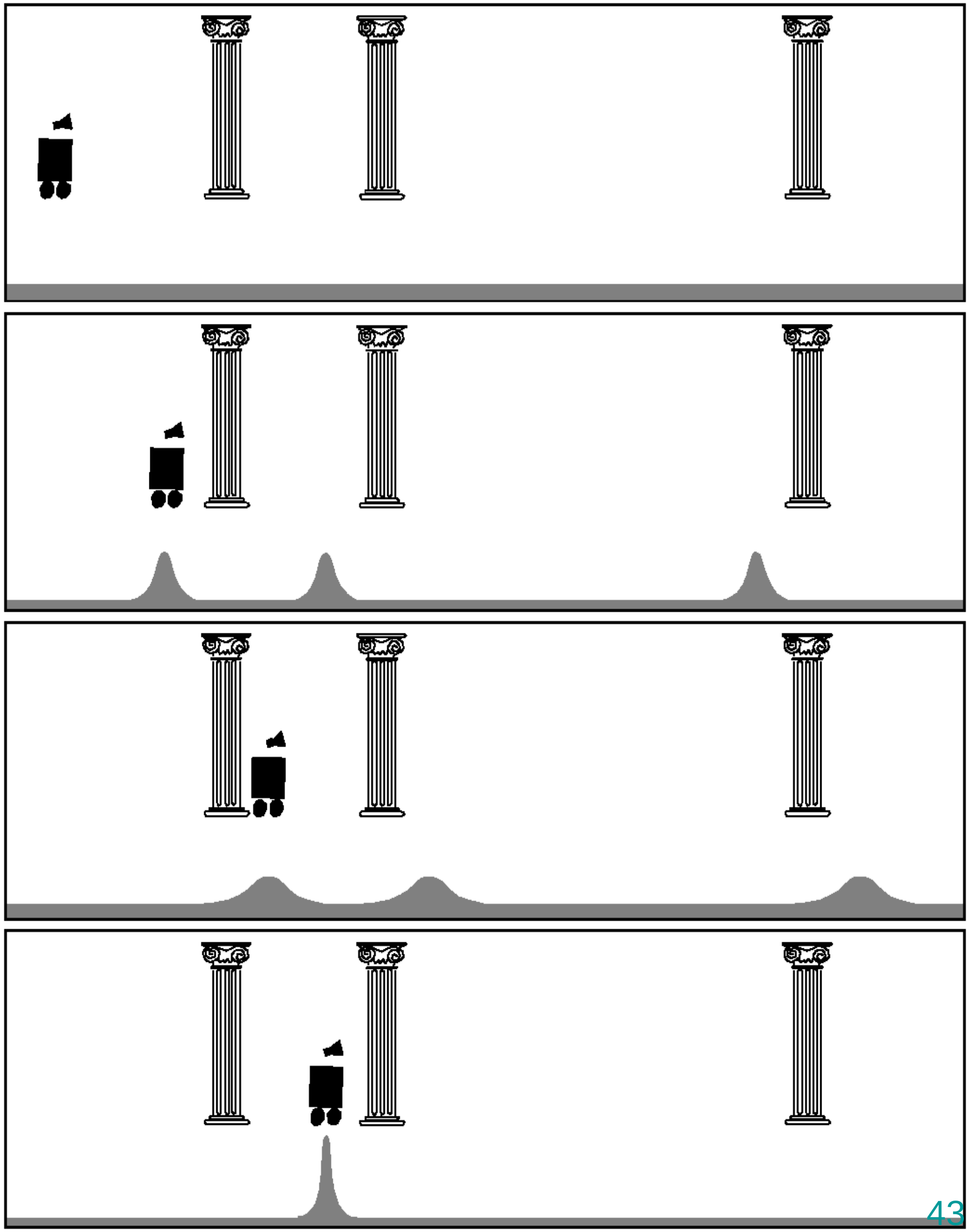
- Perception update
  - perception model SEE

$$s_t = See(i_t, s'_t)$$

  with $\mathbf{i_t}$: exteroceptive sensor inputs, $\mathbf{s'_1}$: updated belief state

  - decreases uncertainty

- *Improving* belief state by ***moving***

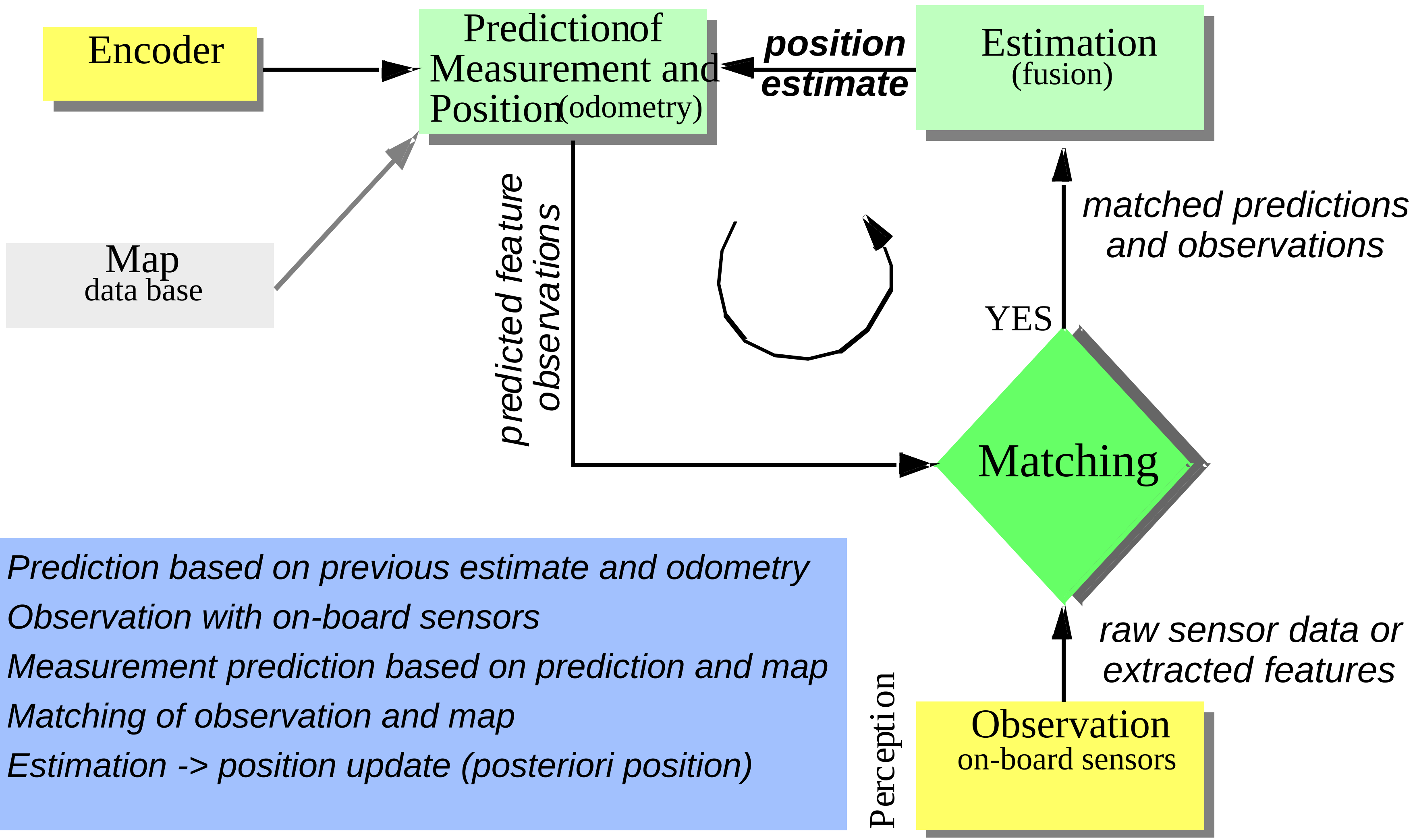


Probabilistic, Map-Based Localization

- Given
  - the position estimate $p(k|k)$
  - its covariance $\Sigma_p(k|k)$ for time k,
  - the current control input $u(k)$
  - the current set of observations $Z(k+1)$ and
  - the map $M(k)$
- Compute the
  - new (posteriori) position estimate $p(k+1|k+1)$ and
  - its covariance $\Sigma_p(k+1|k+1)$

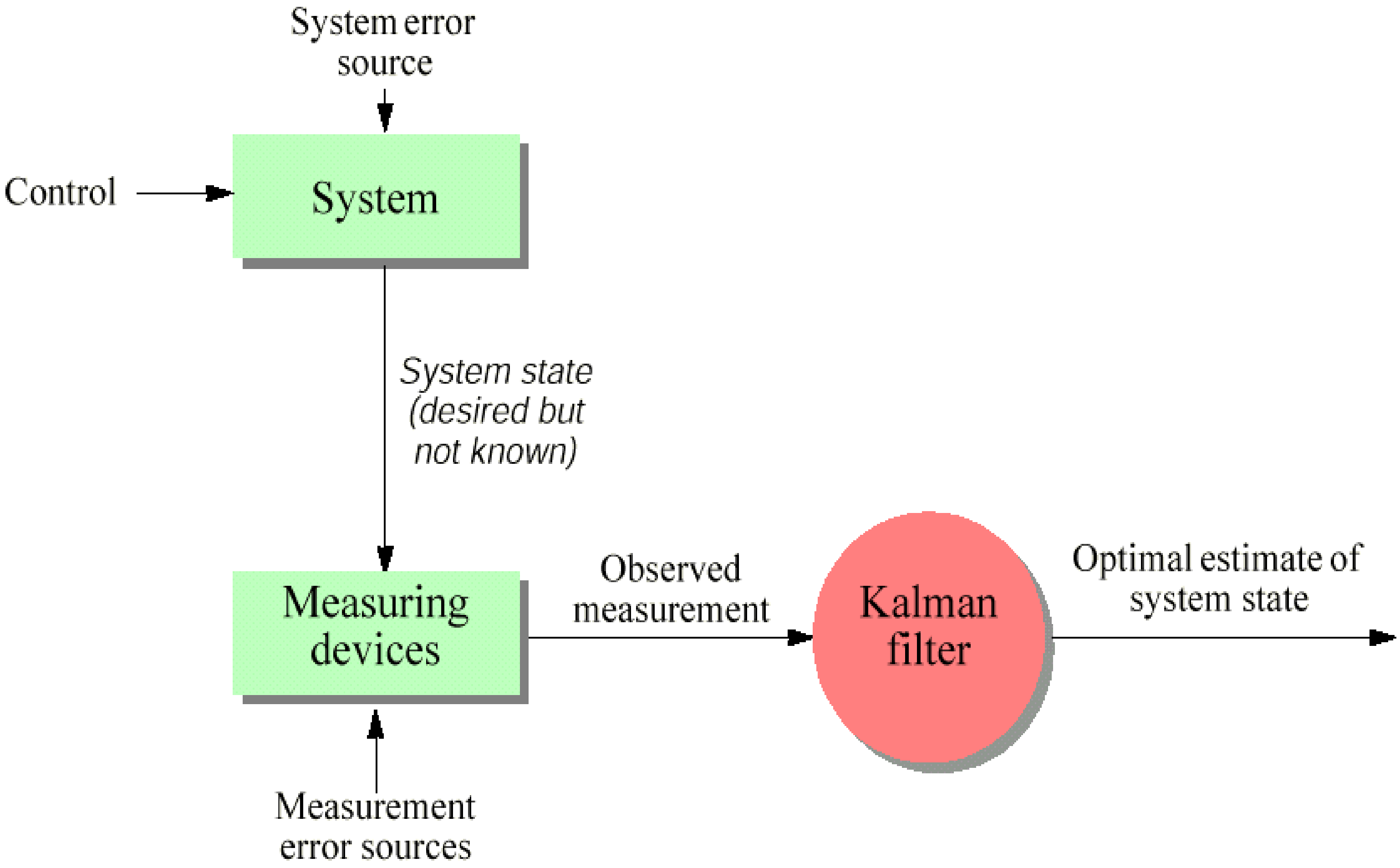
- Such a procedure usually involves five steps:

The Five Steps for Map-Based Localization

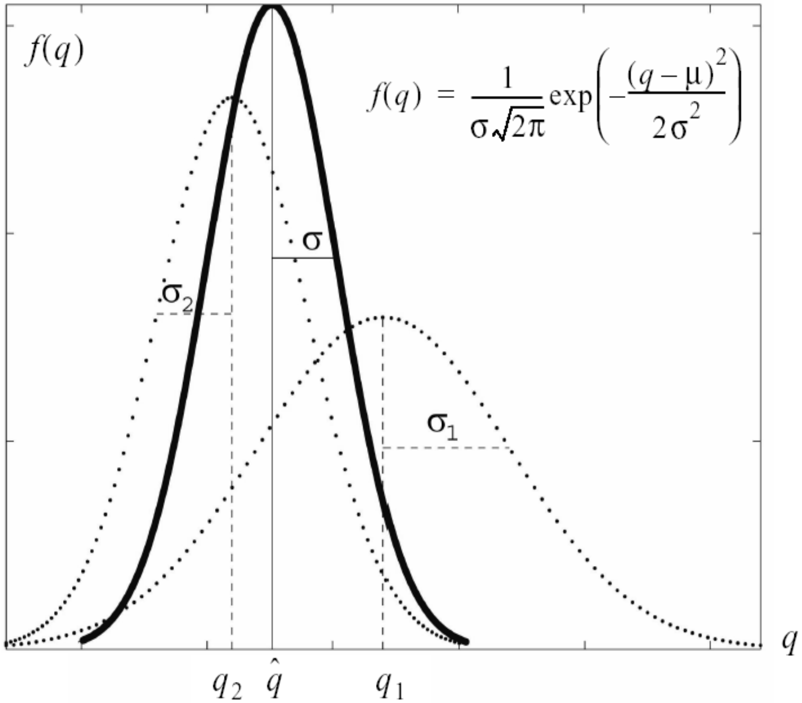


1. *Prediction based on previous estimate and odometry*
2. *Observation with on-board sensors*
3. *Measurement prediction based on prediction and map*
4. *Matching of observation and map*
5. *Estimation -> position update (posteriori position)*

Kalman Filter Localization



Introduction to Kalman Filter



- Two measurements

$$\hat{q}_1 = q_1 \text{ with variance } \sigma_1^2$$

$$\hat{q}_2 = q_2 \text{ with variance } \sigma_2^2$$

- Weighted leas-square

$$S = \sum_{i=1}^{n} w_i(\hat{q} - q_i)^2$$

- Finding minimum error

$$\frac{\partial S}{\partial \hat{q}} = \frac{\partial}{\partial \hat{q}} \sum_{i=1}^{n} w_i(\hat{q} - q_i)^2 = 2\sum_{i=1}^{n} w_i(\hat{q} - q_i) = 0$$

- After some calculation and rearrangements

$$\hat{q} = q_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}(q_2 - q_1)$$

Introduction to Kalman Filter

- In Kalman Filter notation

$$\hat{x}_{k+1} = \hat{x}_k + K_{k+1}(z_{k+1} - \hat{x}_k)$$

$$K_{k+1} = \frac{\sigma_k^2}{\sigma_k^2 + \sigma_z^2} \;\; ; \;\; \sigma_k^2 = \sigma_1^2 \;\; ; \;\; \sigma_z^2 = \sigma_2^2$$

$$\sigma_{k+1}^2 = \sigma_k^2 - K_{k+1}\sigma_k^2$$

Introduction to Kalman Filter

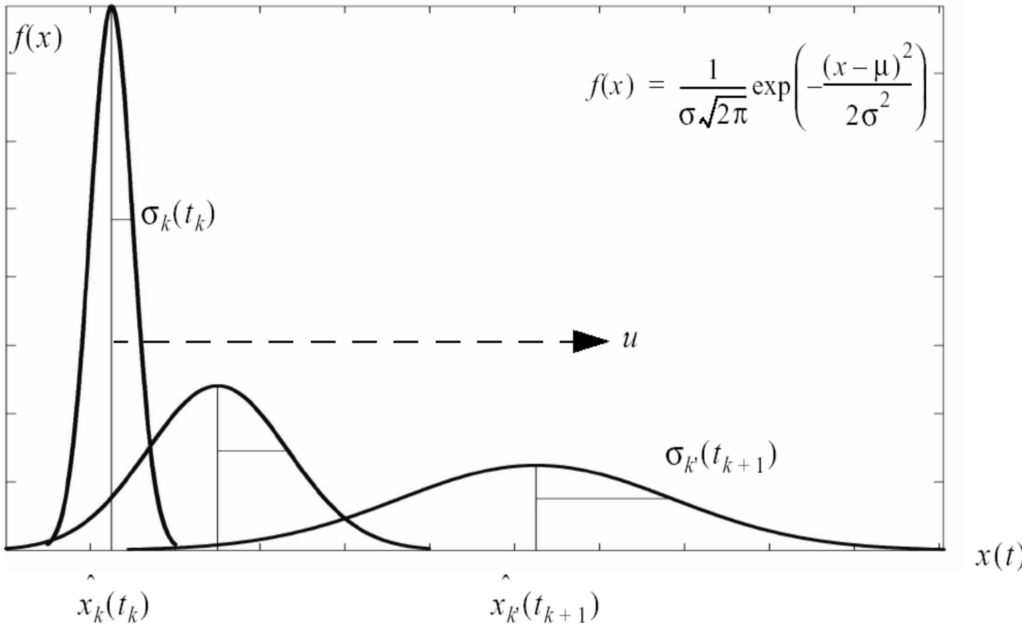
- Dynamic Prediction

  (robot moving)

$$\frac{dx}{dt} = u + w$$

u = velocity

w = noise



- Motion

$$\hat{x}_{k'} = \hat{x}_k + u(t_{k+1} - t_k)$$

$$\sigma_{k'}^2 = \sigma_k^2 + \sigma_w^2[t_{k+1} - t_k]$$

- Combining fusion and dynamic prediction

$$\begin{aligned}\hat{x}_{k+1} &= \hat{x}_{k'} + K_{k+1}(z_{k+1} - \hat{x}_{k'}) \\ &= [\hat{x}_k + u(t_{k+1} - t_k)] + K_{k+1}[z_{k+1} - \hat{x}_k - u(t_{k+1} - t_k)]\end{aligned}$$

$$K_{k+1} = \frac{\sigma_{k'}^2}{\sigma_{k'}^2 + \sigma_z^2} = \frac{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k]}{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k] + \sigma_z^2}$$

Robot Position Prediction

- In a first step, the robots position at time step *k*+1 is predicted based on its old location (time step *k*) and its movement due to the control input *u*(*k*):

$$\hat{p}(k+1|k) = f(\hat{p}(k|k), u(k))$$

f: Odometry function

$$\Sigma_p(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u f^T$$

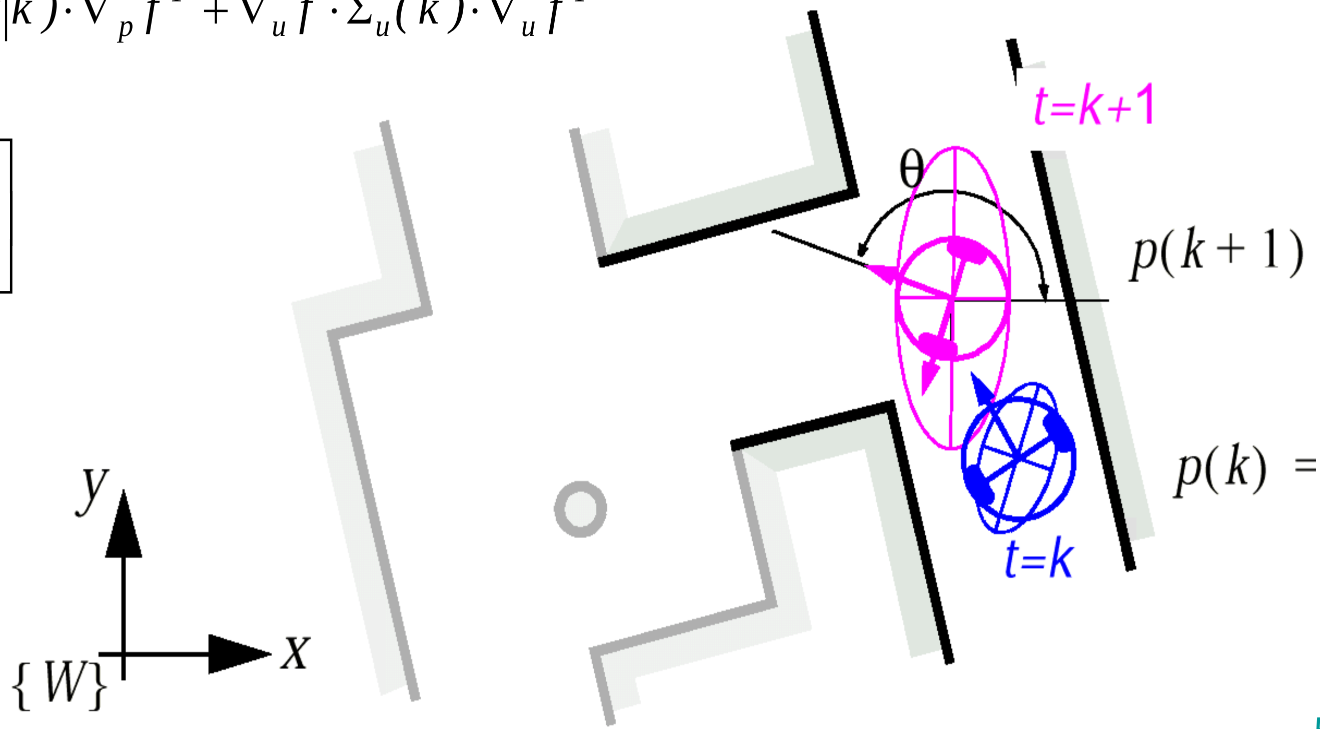
Robot Position Prediction: *Example*

$$\hat{p}(k+1|k) = \hat{p}(k|k) + u(k) = \begin{bmatrix} \hat{x}(k) \\ \hat{y}(k) \\ \hat{\theta}(k) \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2}\cos(\theta + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r + \Delta s_l}{2}\sin(\theta + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

Odometry

$$\Sigma_p(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u f^T$$

$$\Sigma_u(k) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$$

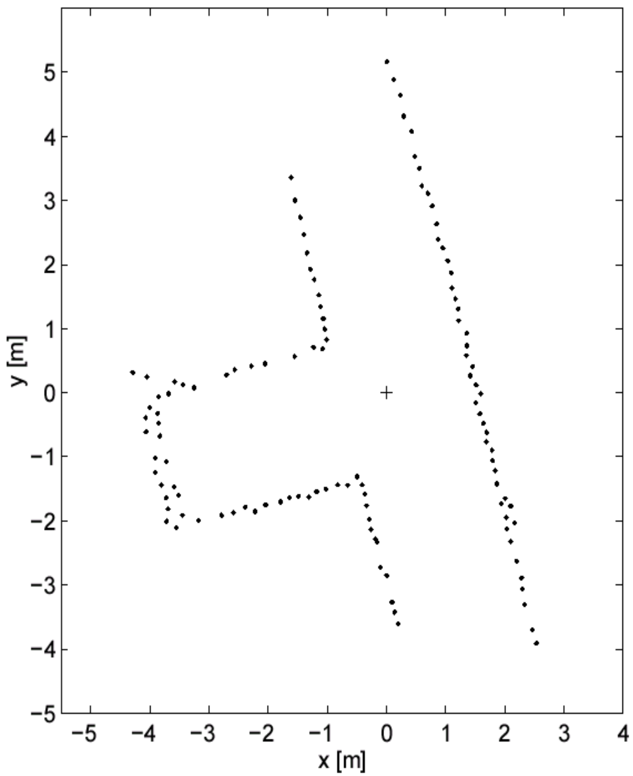


Observation

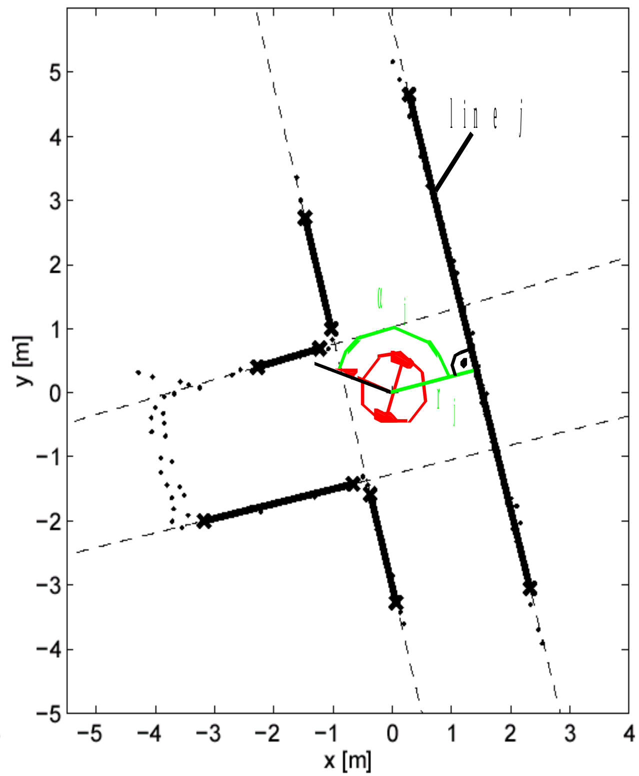
- The second step it to obtain the observation $Z(k+1)$ (measurements) from the robot's sensors at the new location at time $k+1$
- The observation usually consists of a set n_0 of single observations $z_j(k+1)$ extracted from the different sensors signals. It can represent *raw data scans* as well as *features* like *lines*, *doors* or *any kind of landmarks*.
- The parameters of the targets are usually observed in the sensor frame {S}.
  - Therefore the observations have to be transformed to the world frame {W} or
  - the measurement prediction have to be transformed to the sensor frame {S}.
  - This transformation is specified in the function h_i (seen later).

Observation: *Example*

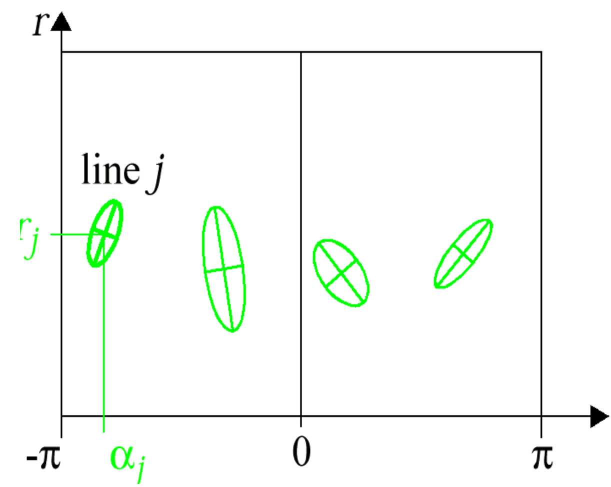
Raw Date of Laser Scanner



Extracted Lines



Extracted Lines in Model Space



$$z_j(k+1) = {}^{R}\begin{bmatrix} \alpha_j \\ r_j \end{bmatrix}$$

Sensor (robot) frame

$$\Sigma_{R,j}(k+1) = \begin{bmatrix} \sigma_{\alpha\alpha} & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_{rr} \end{bmatrix}_j$$

Measurement Prediction

- In the next step we use the predicted robot position $\hat{p} = (k+1|k)$ and the map $M(k)$ to generate multiple predicted observations z_t.
- They have to be transformed into the sensor frame

$$\hat{z}_i(k+1) = h_i\left(z_t, \hat{p}(k+1|k)\right)$$

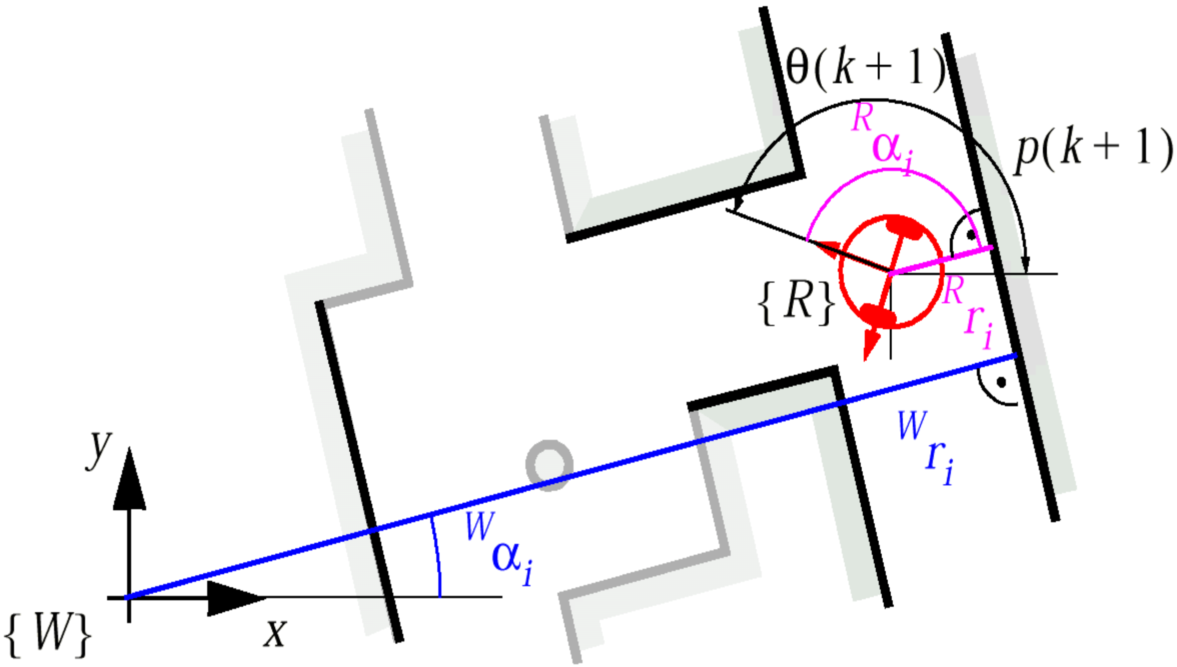
- We can now define the measurement prediction as the set containing all n_i predicted observations

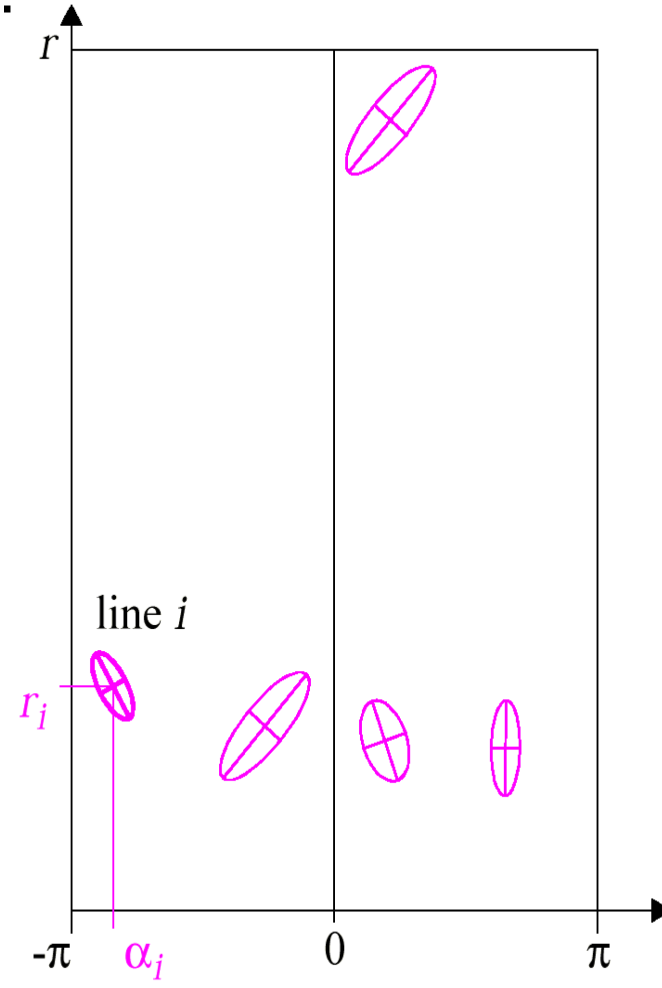
$$\hat{Z}(k+1) = \left\{\hat{z}_i(k+1) \middle| (1 \le i \le n_i)\right\}$$

- The function h_i is mainly the coordinate transformation between the world frame and the sensor frame

Measurement Prediction: *Example*

- For prediction, only the walls that are in the field of view of the robot are selected.
- This is done by linking the individual lines to the nodes of the path





Measurement Prediction: *Example*

- The generated measurement predictions have to be transformed to the robot frame {*R*}

$$ {}^{W}z_{t,i} = {}^{W}\begin{bmatrix} \alpha_{t,i} \\ r_{t,i} \end{bmatrix} \rightarrow {}^{R}z_{t,i} = {}^{R}\begin{bmatrix} \alpha_{t,i} \\ r_{t,i} \end{bmatrix} $$

- According to the figure in previous slide the transformation is given by

$$ \hat{z}_i(k+1) = {}^{R}\begin{bmatrix} \alpha_{t,i} \\ r_{t,i} \end{bmatrix} = h_i(z_{t,i}, \hat{p}(k+1|k)) = \begin{bmatrix} {}^{W}\alpha_{t,i} - {}^{W}\hat{\theta}(k+1|k) \\ {}^{W}r_{t,i} - ({}^{W}\hat{x}(k+1|k)\cos({}^{W}\alpha_{t,i}) + {}^{W}\hat{y}(k+1|k)\sin({}^{W}\alpha_{t,i})) \end{bmatrix} $$

and its Jacobian by

$$ \nabla h_i = \begin{bmatrix} \frac{\partial\alpha_{t,i}}{\partial\hat{x}} & \frac{\partial\alpha_{t,i}}{\partial\hat{y}} & \frac{\partial\alpha_{t,i}}{\partial\hat{\theta}} \\ \frac{\partial r_{t,i}}{\partial\hat{x}} & \frac{\partial r_{t,i}}{\partial\hat{y}} & \frac{\partial r_{t,i}}{\partial\hat{\theta}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ -\cos{}^{W}\alpha_{t,i} & -\sin{}^{W}\alpha_{t,i} & 0 \end{bmatrix} $$

Matching

- Assignment from observations $z_j(k+1)$ (gained by the sensors) to the targets z_t (stored in the map)
- For each measurement prediction for which a corresponding observation is found we calculate the innovation:

$$v_{ij}(k+1) = [z_j(k+1) - h_i(z_t, \hat{p}(k+1|k))]$$

$$= \begin{bmatrix} \alpha_j \\ r_j \end{bmatrix} - \begin{bmatrix} {}^W\alpha_{t,i} - {}^W\hat{\theta}(k+1|k) \\ {}^W r_{t,i} - ({}^W\hat{x}(k+1|k)\cos({}^W\alpha_{t,i}) + {}^W\hat{y}(k+1|k)\sin({}^W\alpha_{t,i})) \end{bmatrix}$$

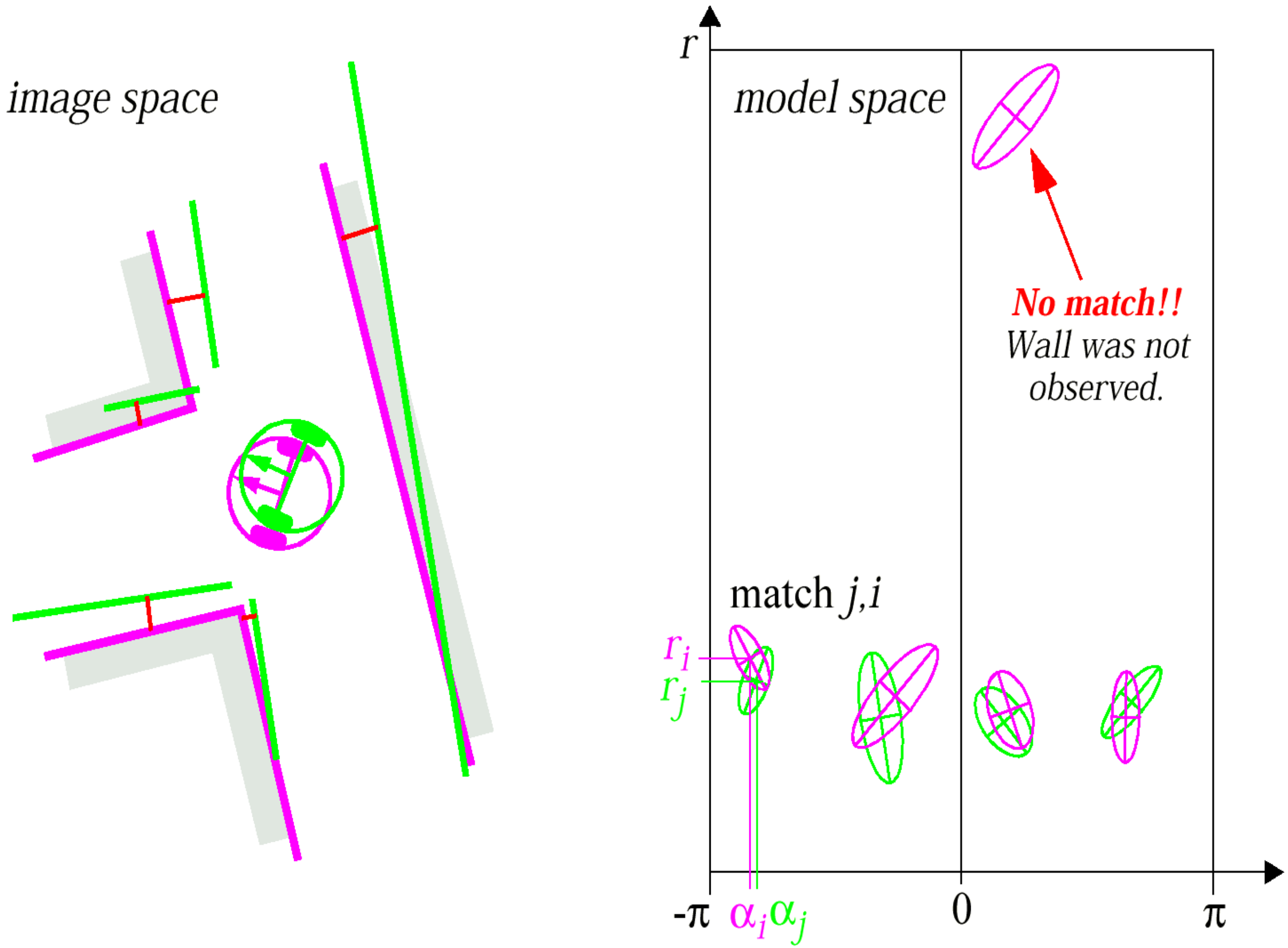
and its innovation covariance found by applying the error propagation law:

$$\Sigma_{IN,ij}(k+1) = \nabla h_i \cdot \Sigma_p(k+1|k) \cdot \nabla h_i^{\,T} + \Sigma_{R,i}(k+1)$$

- The validity of the correspondence between measurement and prediction can e.g. be evaluated through ***the Mahalanobis distance***:

$$v_{ij}^T(k+1) \cdot \Sigma_{IN,ij}^{-1}(k+1) \cdot v_{ij}(k+1) \le g^2$$

Matching: *Example*



Matching: *Example*

- To find correspondence (pairs) of predicted and observed features we use the Mahalanobis distance

$$v_{ij}(k+1)\cdot\Sigma_{IN,\,ij}^{-1}(k+1)\cdot v_{ij}^{T}(k+1)\le g^{2}$$

with

$$v_{ij}(k+1) = [z_j(k+1) - h_i(z_t, \hat{p}(k+1|k))]$$

$$= \begin{bmatrix}\alpha_j \\ r_j\end{bmatrix} - \begin{bmatrix} {}^{W}\alpha_{t,i} - {}^{W}\hat{\theta}(k+1|k) \\ {}^{W}r_{t,i} - ({}^{W}\hat{x}(k+1|k)\cos({}^{W}\alpha_{t,i}) + {}^{W}\hat{y}(k+1|k)\sin({}^{W}\alpha_{t,i})) \end{bmatrix}$$

$$\Sigma_{IN,\,ij}(k+1) = \nabla h_i\cdot\Sigma_p(k+1|k)\cdot\nabla h_i^{\,T} + \Sigma_{R,\,i}(k+1)$$

Estimation: Applying the Kalman Filter

- Kalman filter gain:

$$K(k+1) = \Sigma_p(k+1|k) \cdot \nabla h^T \cdot \Sigma_{IN}^{-1}(k+1)$$

- Update of robot's position estimate:

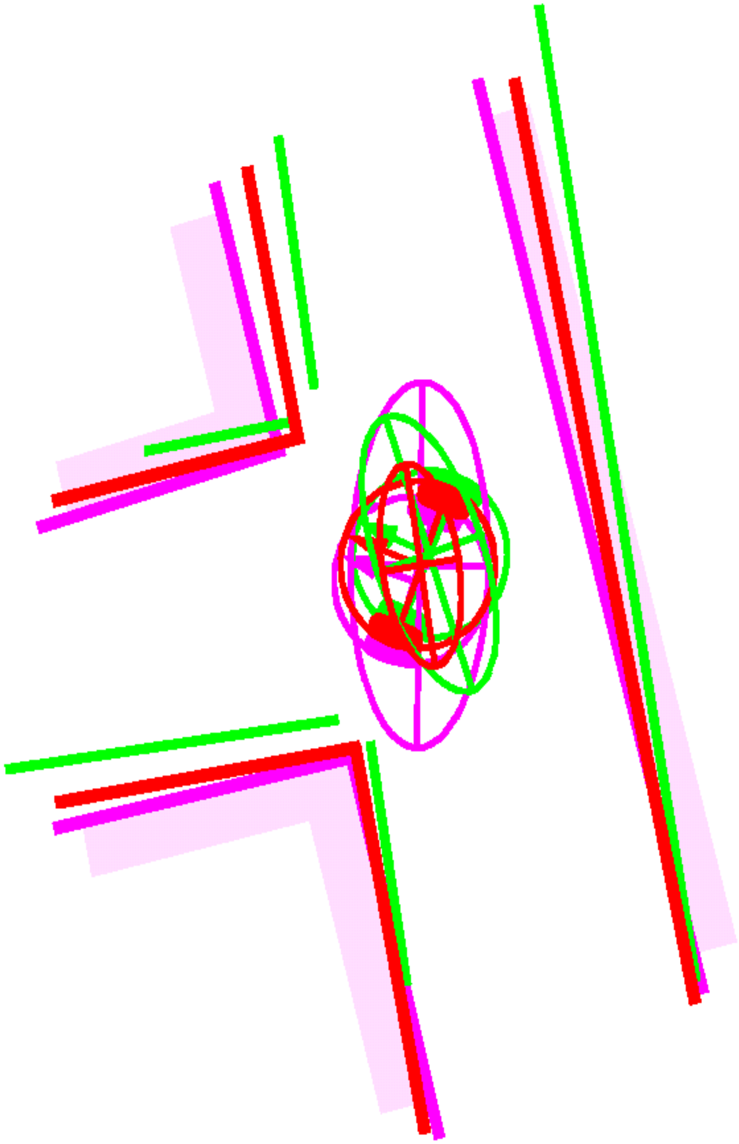
$$\hat{p}(k+1|k+1) = \hat{p}(k+1|k) + K(k+1) \cdot v(k+1)$$

- The associate variance

$$\Sigma_p(k+1|k+1) = \Sigma_p(k+1|k) - K(k+1) \cdot \Sigma_{IN}(k+1) \cdot K^T(k+1)$$

Estimation: *Example*

- Kalman filter estimation of the new robot position $\hat{p}(k|k)$:
  - By fusing the prediction of robot position (magenta) with the innovation gained by the measurements (green) we get the updated estimate of the robot position (red)



Markov Localization

- Markov localization uses an explicit, discrete representation for the probability of all position in the state space.
- This is usually done by representing the environment by a grid or a topological graph with a finite number of possible states (positions).
- During each update, the probability for each state (element) of the entire space is updated.

Markov Localization

Applying probability theory to robot localization

- P(A): Probability that A is true.
  - e.g. $p(r_t = l)$: probability that the robot r is at position l at time t
- We wish to compute the probability of each individual robot position given actions and sensor measures.
- P(A|B): Conditional probability of A given that we know B.
  - e.g. $p(r_t = l | i_t)$: probability that the robot is at position l given the sensors input i_t.
- Product rule:

$$p(A \wedge B) = p(A|B)p(B) \qquad p(A \wedge B) = p(B|A)p(A)$$

- Bayes rule: $$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

Markov Localization

- Bayes rule:

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

  - Map from a belief state and a sensor input to a refined belief state (SEE):

$$p(l|i) = \frac{p(i|l)p(l)}{p(i)}$$

  - p(l): belief state before perceptual update process
  - p(i |l): probability to get measurement i when being at position l
    - consult robots map, identify the probability of a certain sensor reading for each possible position in the map
  - p(i): normalization factor so that sum over all l for L equals 1.

Markov Localization

- Bayes rule:

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

  – Map from a belief state and a action to new belief state (ACT):

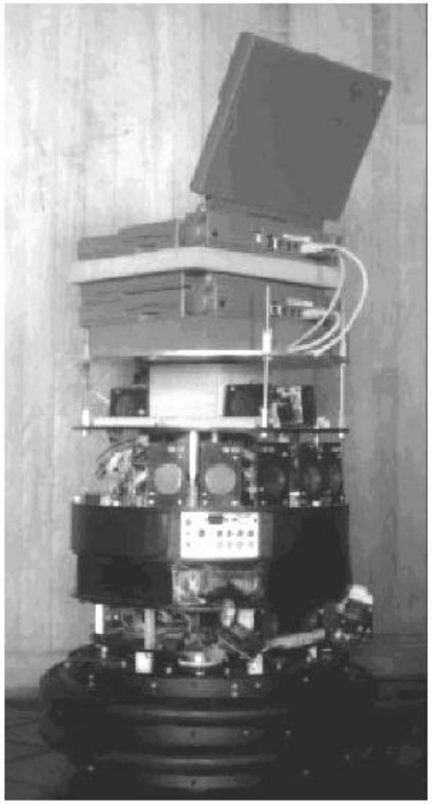
$$p(l_t|o_t) = \int p(l_t|l'_{t-1}, o_t)p(l'_{t-1})dl'_{t-1}$$

  – Summing over all possible ways in which the robot may have reached l.

- Markov assumption: Update only depends on previous state and its most recent actions and perception.

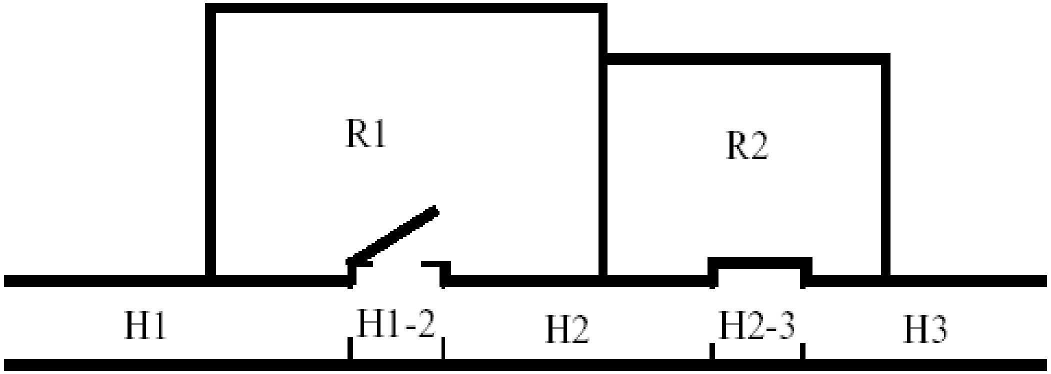
Case Study 1 - Topological Map (1)

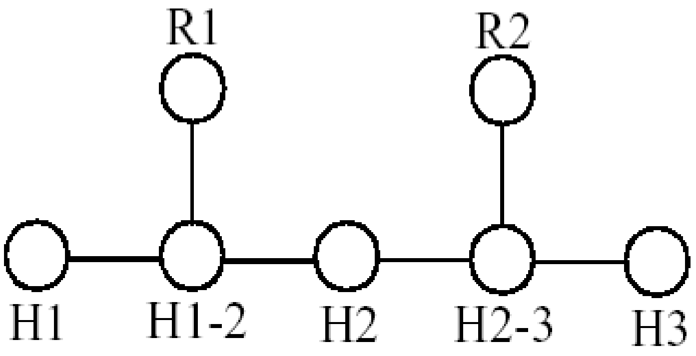
- The Dervish Robot
- Topological Localization with Sonar



Case Study 1 - Topological Map (2)

- Topological map of office-type environment





	Wall	Closed door	Open door	Open hallway	Foyer
Nothing detected	0.70	0.40	0.05	0.001	0.30
Closed door detected	0.30	0.60	0	0	0.05
Open door detected	0	0	0.90	0.10	0.15
Open hallway detected	0	0	0.001	0.90	0.50

Case Study 1 - Topological Map (3)

- Update of believe state for position *n* given the percept-pair *i*

$$p(n|i) = p(i|n)p(n)$$

  - p(n¦i): new likelihood for being in position *n*
  - p(n): current believe state
  - p(i¦n): probability of seeing *i* in *n* (see table)

	Wall	Closed door	Open door	Open hallway	Foyer
Nothing detected	0.70	0.40	0.05	0.001	0.30
Closed door detected	0.30	0.60	0	0	0.05
Open door detected	0	0	0.90	0.10	0.15
Open hallway detected	0	0	0.001	0.90	0.50

- No action update !
  - However, the robot is moving and therefore we can apply a combination of action and perception update

$$p(n_t|i_t) = \int p(n_t|n'_{t-i}, i_t)p(n'_{t-i})dn'_{t-i}$$

  - t-i is used instead of t-1 because the topological distance between n' and n can vary depending on the specific topological map

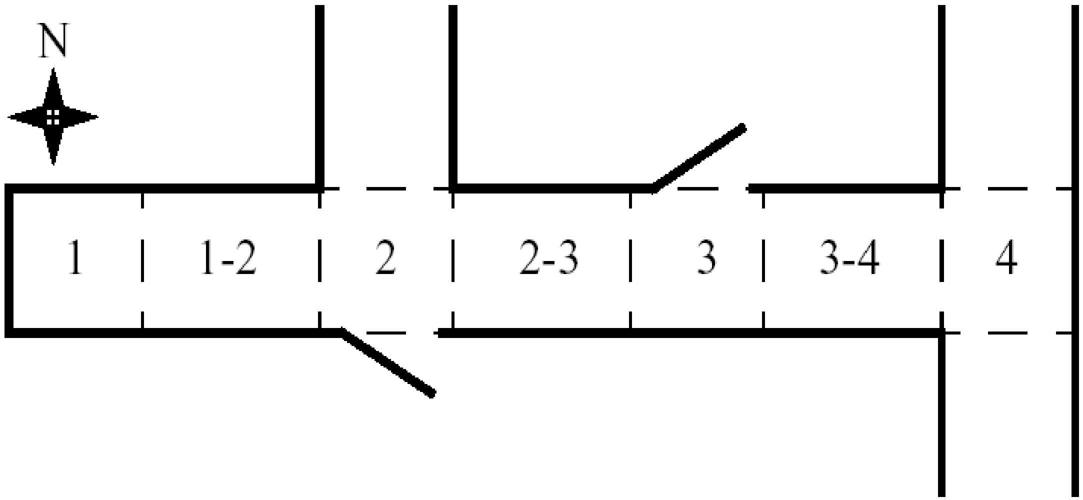
Case Study 1 - Topological Map (4)

- The calculation

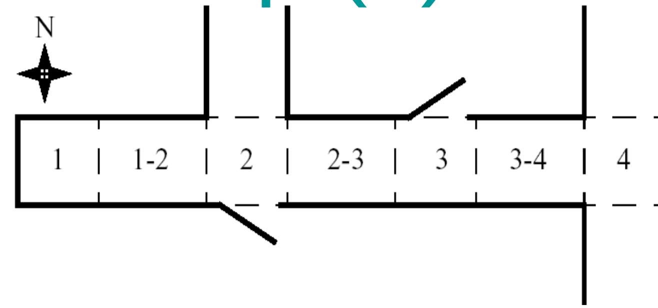
  $$p(n_t | n'_{t-i}, i_t)$$

  is calculated by multiplying the probability of generating perceptual event *i* at position *n* by the probability of having failed to generate perceptual events at all nodes between *n'* and *n*.

$$p(n_t | n'_{t-i}, i_t) = p(i_t, n_t) \cdot p(\varnothing, n_{t-1}) \cdot p(\varnothing, n_{t-2}) \cdot \ldots \cdot p(\varnothing, n_{t-i+1})$$



Case Study 1 - Topological Map (5)



- Example calculation
  - Assume that the robot has two nonzero belief
    - p*(1-2) = 1.0 ;* p*(2-3) = 0.2* *

    at that it is facing east with certainty
  - State *2-3* will progress potentially to *3*, *3-4* t and *4*.
  - State *3* and *3-4* can be eliminated because the likelihood of detecting an open door is zero.
  - The likelihood of reaching state *4* is the product of the initial likelihood p*(2-3)= 0.2*, (a) the likelihood of detecting anything at node *3* and the likelihood of detecting a hallway on the left and a door on the right at node *4* and (b) the likelihood of detecting a hallway on the left and a door on the right at node *4*. (for simplicity we assume that the likelihood of detecting nothing at node *3-4* is *1.0*)
  - This leads to:
    - $0.2 \times [0.6 \times 0.4 + 0.4 \times 0.05] \times 0.7 \times [0.9 \times 0.1] \rightarrow$ p(4) = 0.003.
    - Similar calculation for progress from 1-2 $\rightarrow$ p(2) = 0.3.

** Note that the probabilities do not sum up to one. For simplicity normalization was avoided in this example*

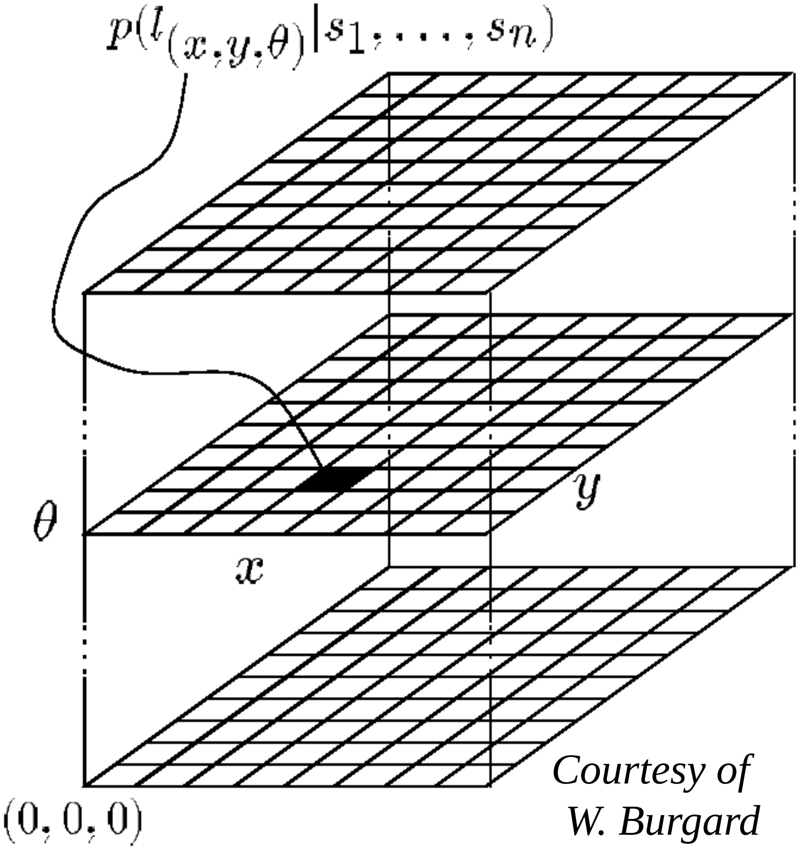
Case Study 2 – Grid Map

- Fine *fixed decomposition* grid (x, y, θ), 15 cm x 15 cm x 1°
  - Action and perception update
- Action update:
  - Sum over previous possible positions and motion model

$$P(l_t|o_t) = \sum_{l'} P(l_t|l'_{t-1}, o_t) \cdot p(l'_{t-1})$$

  - Discrete version of eq. 5.22
- Perception update:
  - Given perception i, what is the probability to be a location l

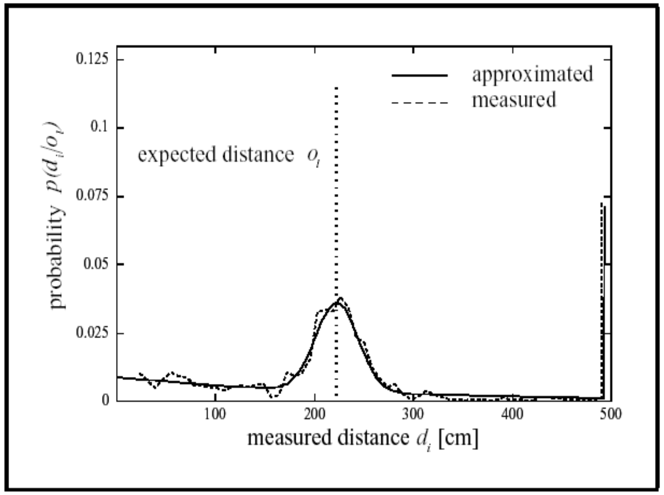
$$p(l|i) = \frac{p(i|l)p(l)}{p(i)}$$



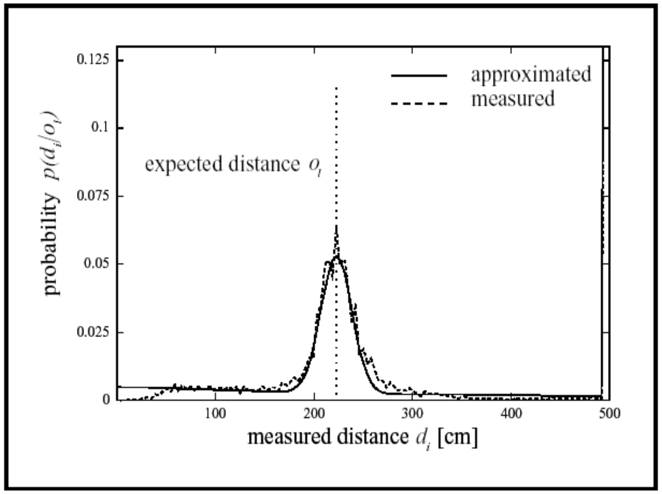
Courtesy of W. Burgard

Case Study 2 – Grid Map

- The critical challenge is the calculation of $p(i|l)$ $$p(l|i) = \frac{p(i|l)p(l)}{p(i)}$$
  - The number of possible sensor readings and geometric contexts is extremely large
  - $p(i|l)$ is computed using a model of the robot's sensor behavior, its position l, and the local environment metric map around l.
  - Assumptions
    - Measurement error can be described by a distribution with a mean
    - Non-zero chance for any measurement



Ultrasound.

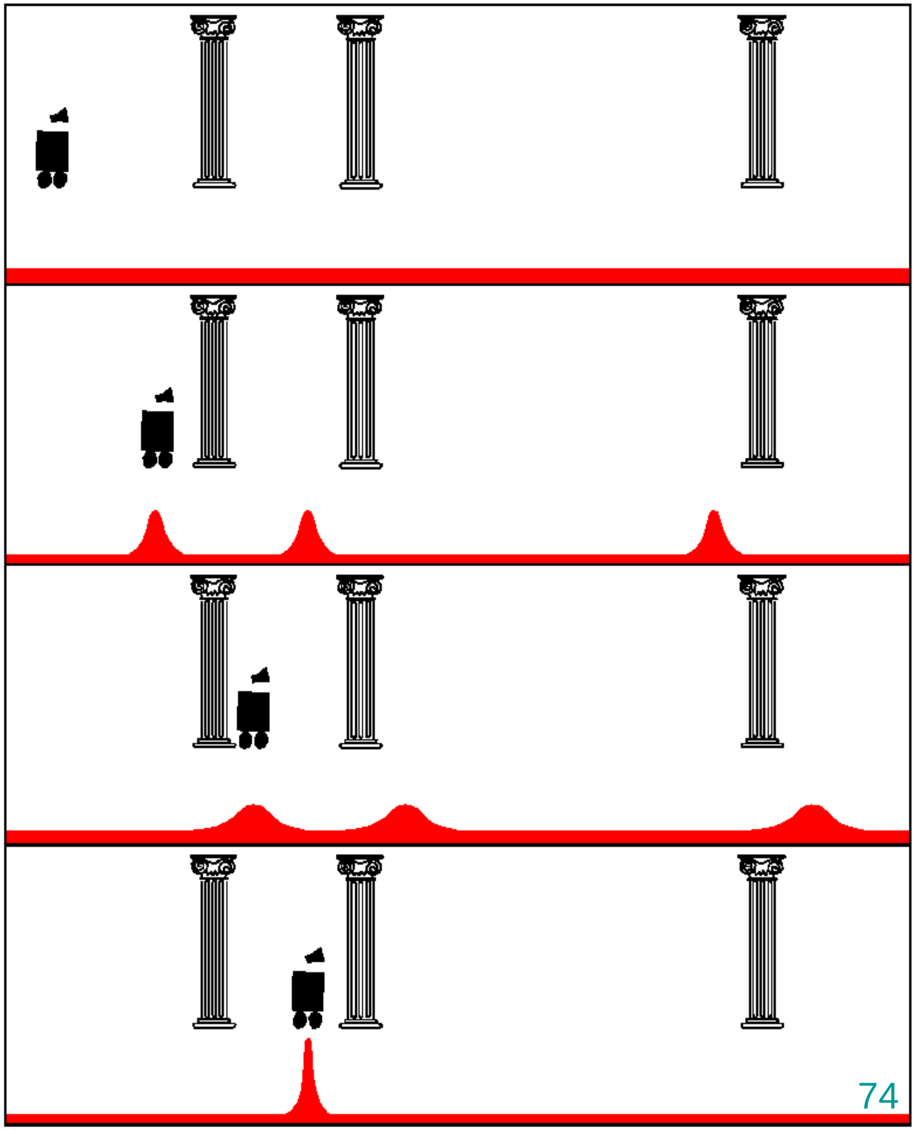


Laser range-finder.

Courtesy of W. Burgard

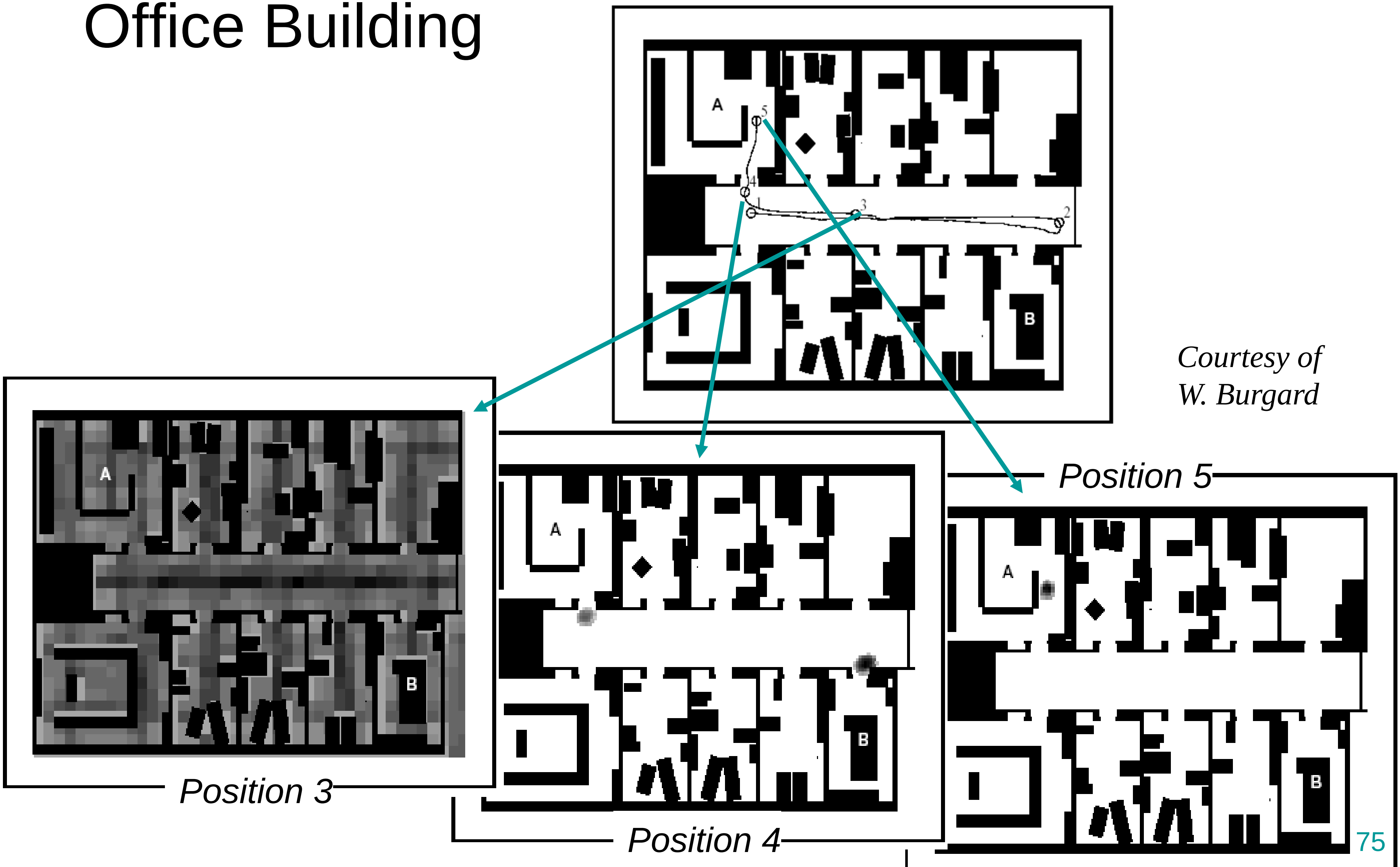
Case Study 2 – Grid Map

- The 1D case
  1. Start
     - No knowledge at start, thus we have an uniform probability distribution.
  2. Robot perceives first pillar
     - Seeing only one pillar, the probability being at pillar 1, 2 or 3 is equal.
  3. Robot moves
     - Action model enables to estimate the new probability distribution based on the previous one and the motion.
  4. Robot perceives second pillar
     - Base on all prior knowledge the probability being at pillar 2 becomes dominant



Case Study 2 – Grid Map

Office Building

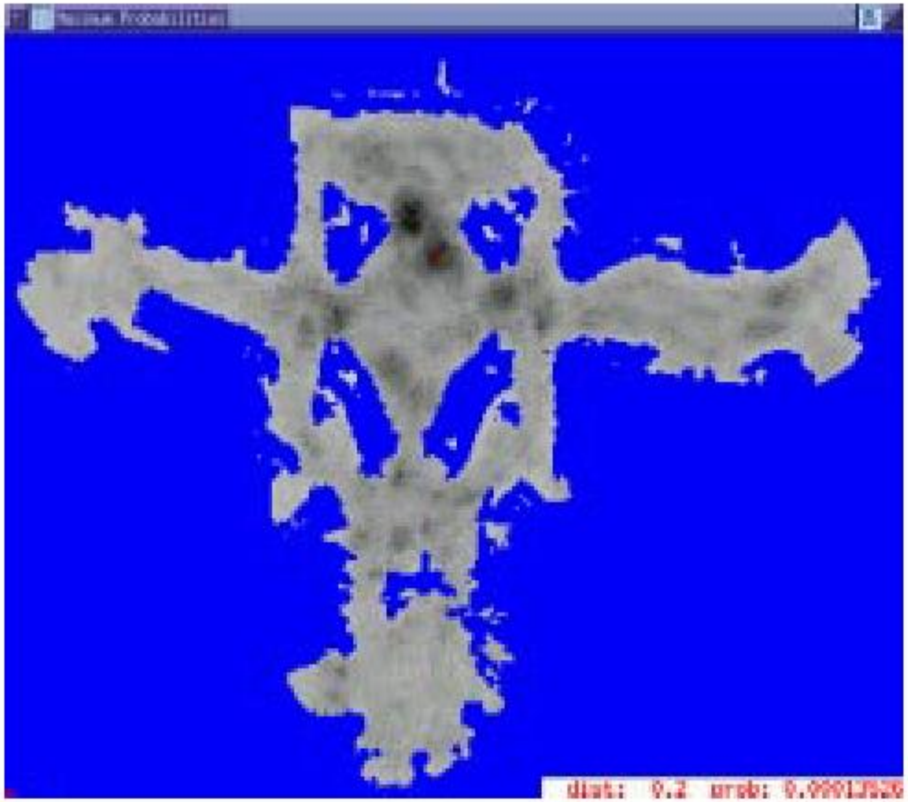


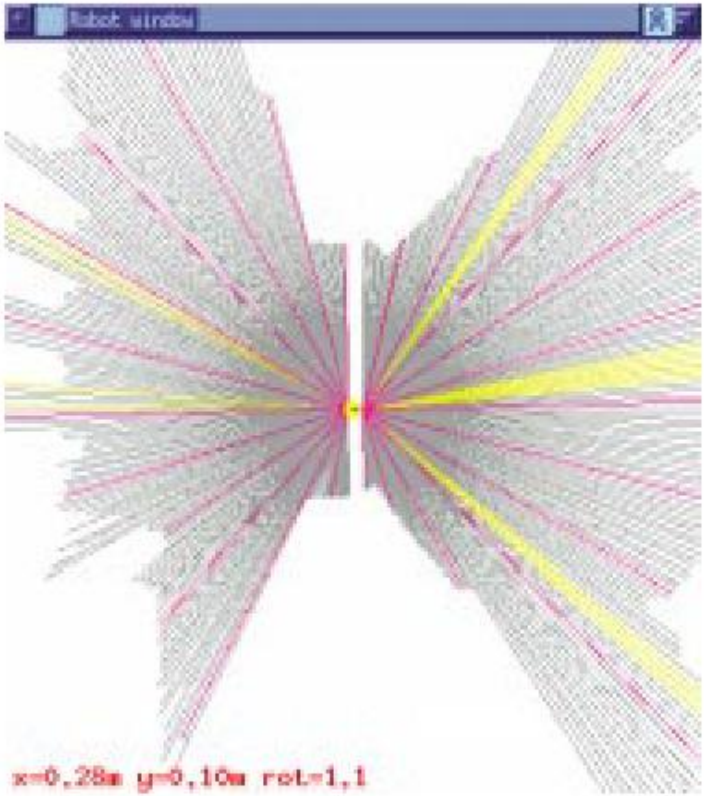
Courtesy of W. Burgard

Case Study 2 – Grid Map

- Example 2: Museum
  - Laser scan 1

Courtesy of
W. Burgard

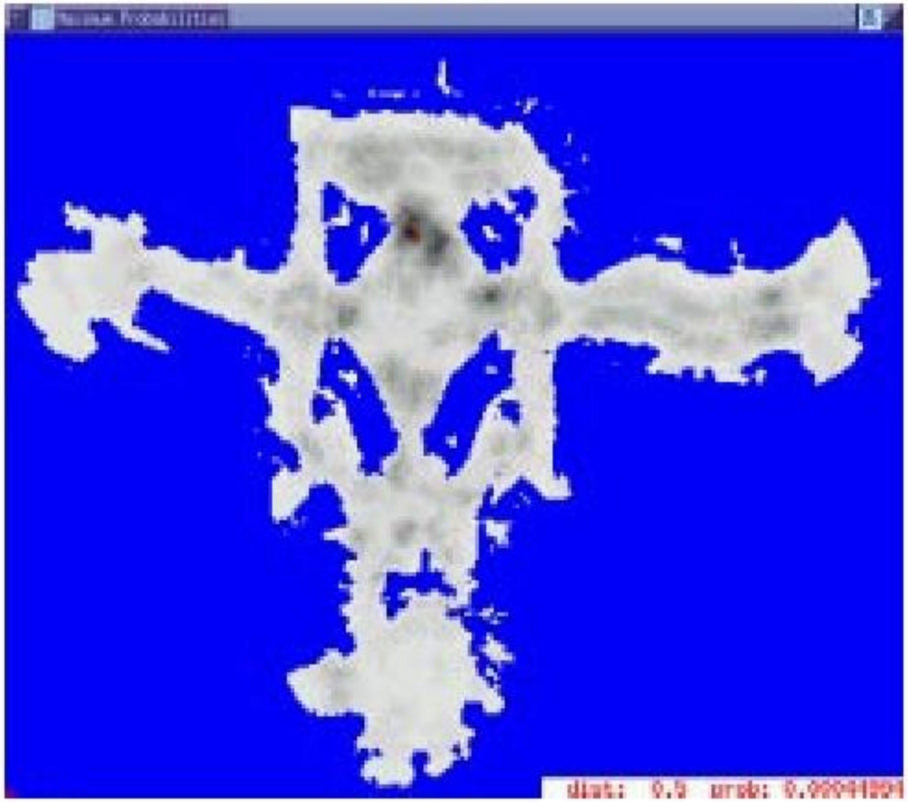


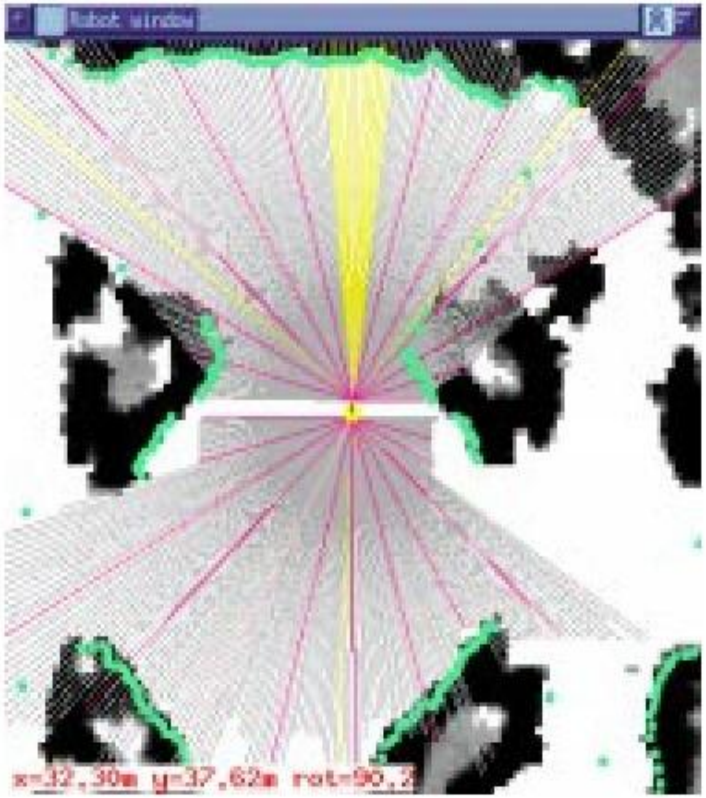


Case Study 2 – Grid Map

- Example 2: Museum
  - Laser scan 2

Courtesy of W. Burgard

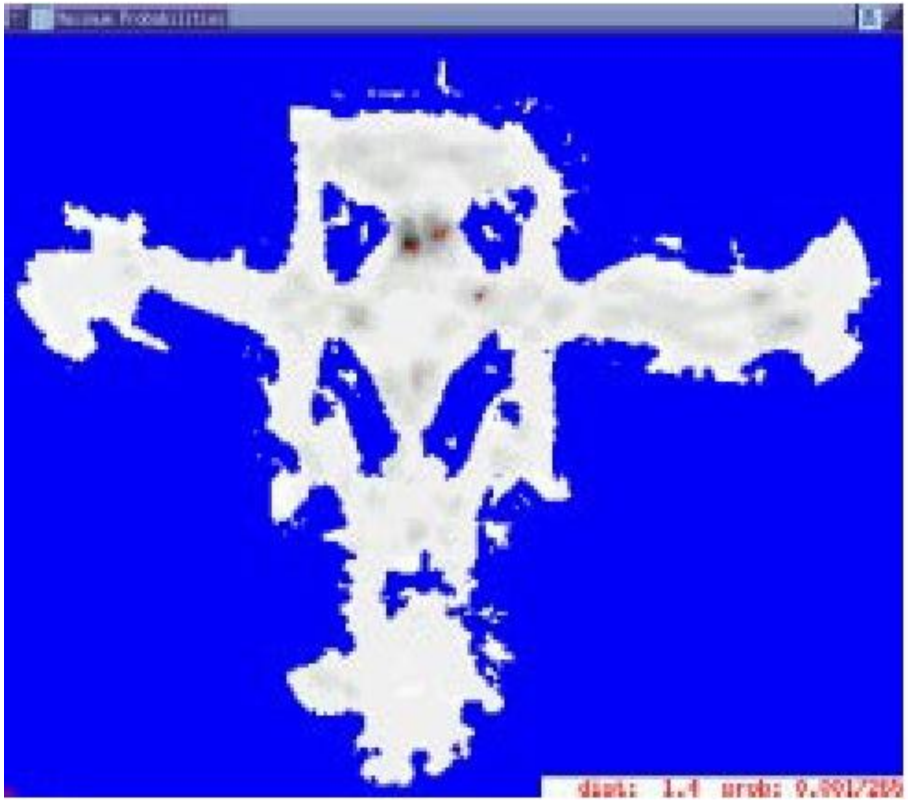


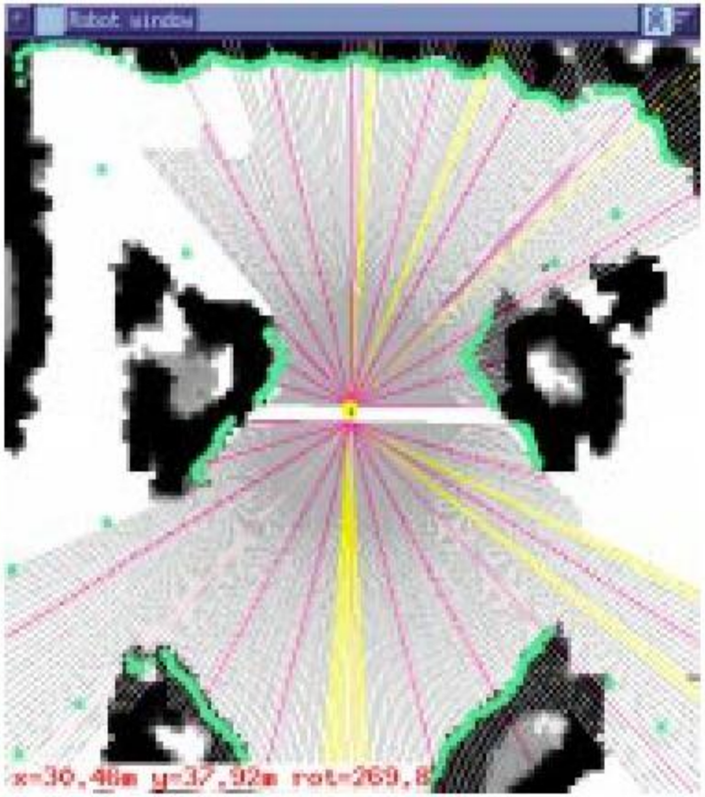


Case Study 2 – Grid Map

- Example 2: Museum
  - Laser scan 3

Courtesy of W. Burgard

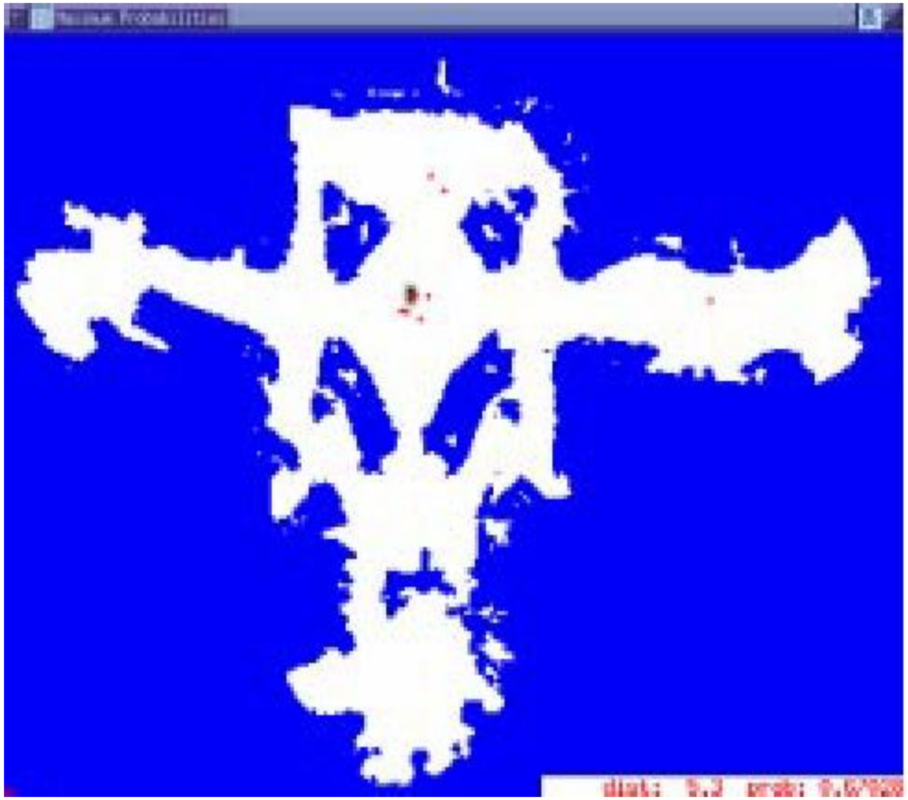


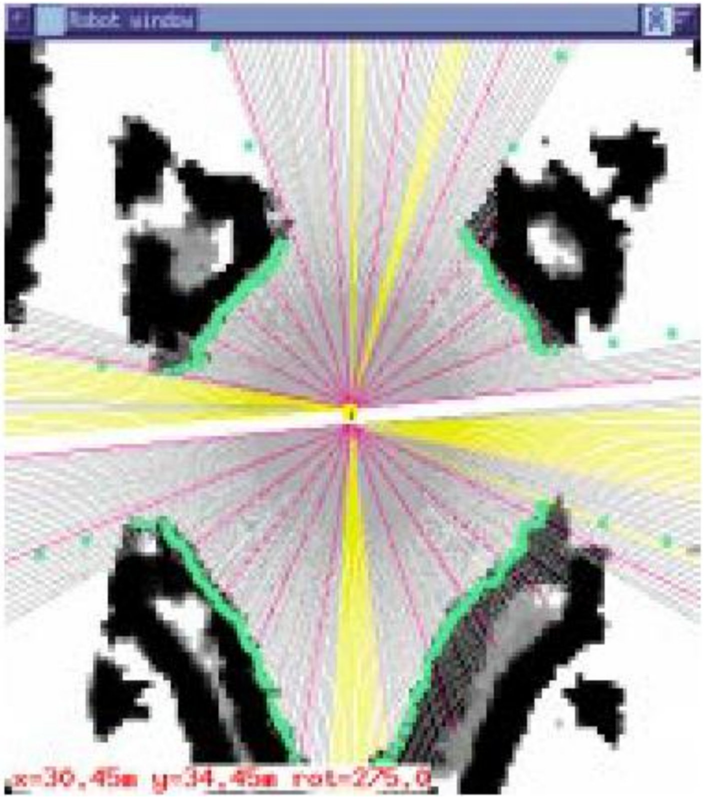


Case Study 2 – Grid Map

Courtesy of W. Burgard

- Example 2: Museum
  - Laser scan 13

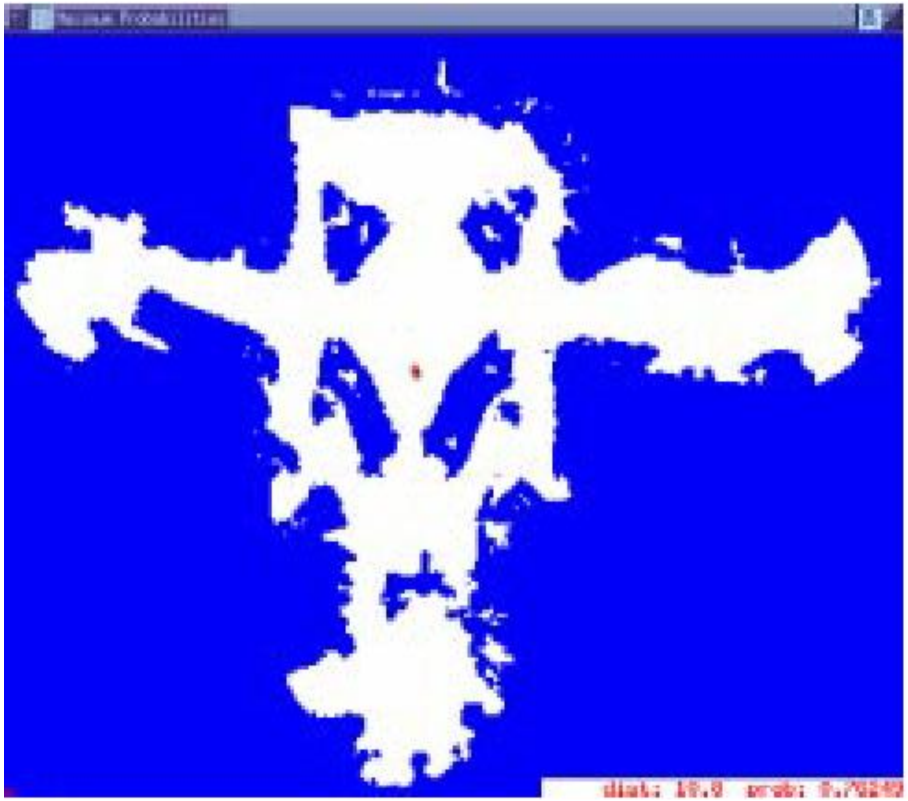


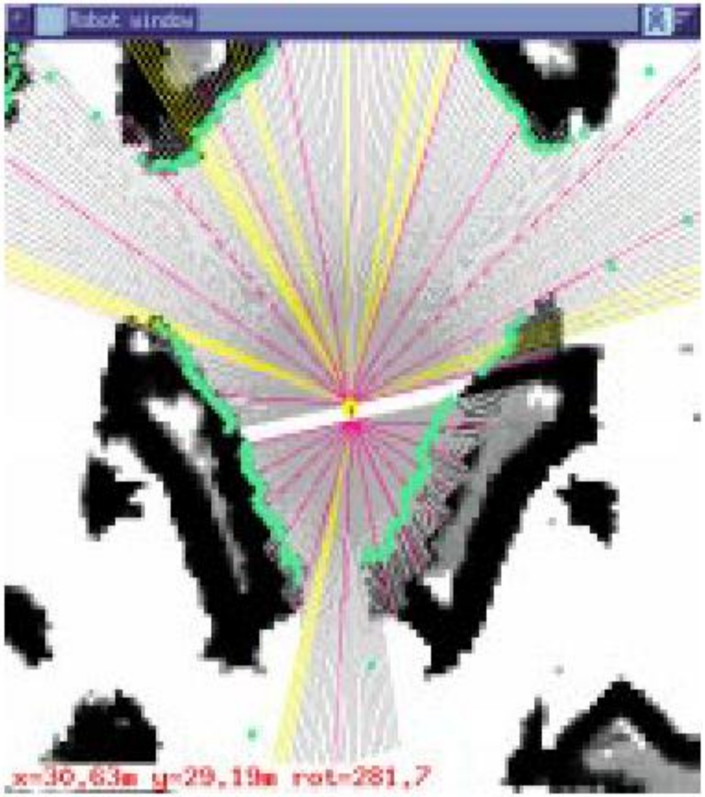


Case Study 2 – Grid Map

- Example 2: Museum
  - Laser scan 21

Courtesy of W. Burgard





Case Study 2 – Grid Map

- Fine *fixed decomposition* grids result in a huge state space
  - Very important processing power needed
  - Large memory requirement
- Reducing complexity
  - Various approached have been proposed for reducing complexity
  - The main goal is to reduce the number of states that are updated in each step
- Randomized Sampling / Particle Filter
  - Approximated belief state by representing only a 'representative' subset of all states (possible locations)
  - E.g update only 10% of all possible locations
  - The sampling process is typically weighted, e.g. put more samples around the local peaks in the probability density function
  - However, you have to ensure some less likely locations are still tracked, otherwise the robot might get lost

Markov ⇔ Kalman Filter Localization

- Markov localization
  - localization starting from any unknown position
  - recovers from ambiguous situation.
  - However, to update the probability of all positions within the whole state space at any time requires a discrete representation of the space (grid). The required memory and calculation power can thus become very important if a fine grid is used.
- Kalman filter localization
  - tracks the robot and is inherently very precise and efficient.
  - However, if the uncertainty of the robot becomes to large (e.g. collision with an object) the Kalman filter will fail and the position is definitively lost.

Questions?