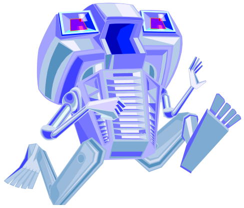
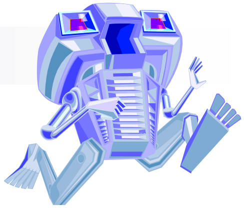


ECE497: Introduction to Mobile Robotics Lecture 4

Dr. Carlotta A. Berry

Spring 06 - 07



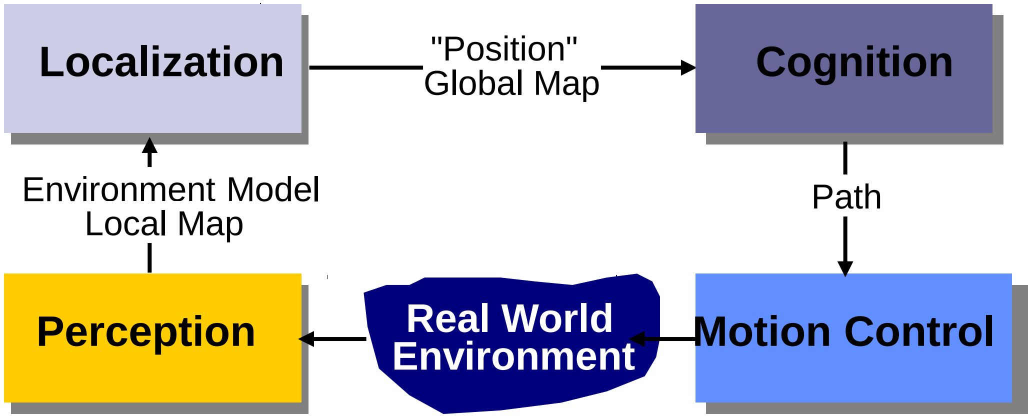
Quote of the Week

“A common mistake people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools.”

D. Adams

Mobile Robot Localization (5.1)

- *Navigation* is one of the most challenging mobile robot competencies
- Successful *navigation* requires
  - □ Perception
  - □ Localization
  - □ Cognition
  - □ Motion control



Localization

"Position"
Global Map

Cognition

Path

Motion Control

Real World
Environment

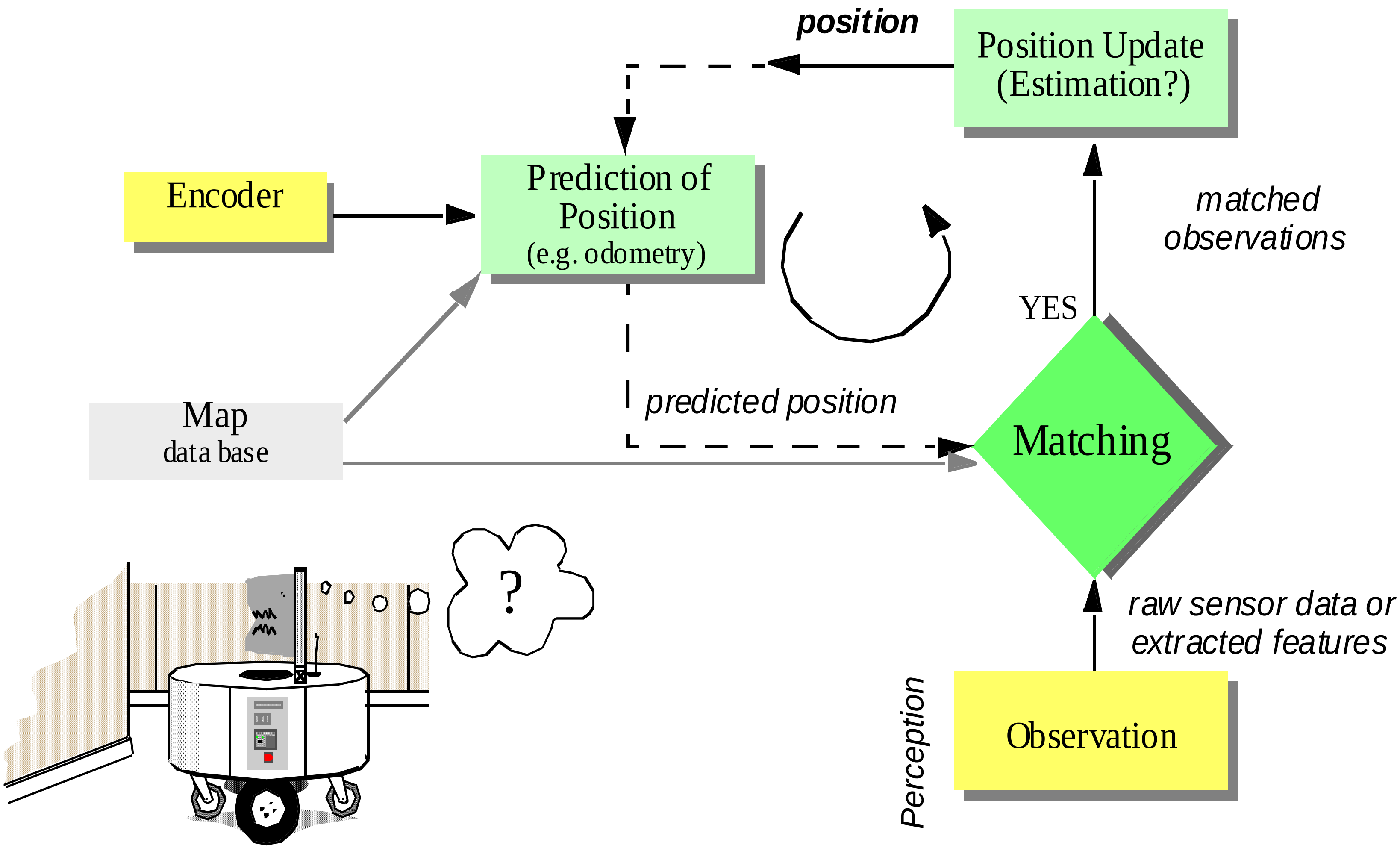
Perception

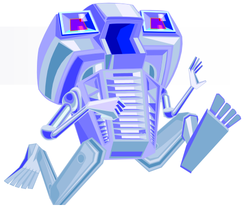
Environment Model
Local Map

Mobile Robot Localization (5.1)

- ***Perception***
  - □ The robot must interprets its sensors to extract meaningful data
- ***Localization***
  - □ The robot must determine it’s position in the environment
- ***Cognition***
  - □ The robot must decide how to act to achiever its goals
- ***Motion Control***
  - □ The robot must modulates its motor outputs to achieve the desired trajectory

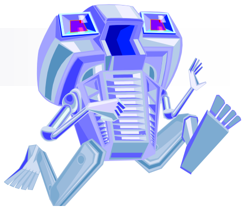
Localization, Where am I? (5.1)





What’s the problem?

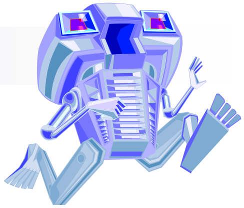
- WHERE AM I?
- But what does this mean, really?
- Frame of reference is important
  - □ Local/Relative: Where am I vs. where I was?
  - □ Global/Absolute: Where am I relative to the world frame?
- Location can be specified in two ways
  - □ Geometric: Distances and angles
  - □ Topological: Connections among landmarks



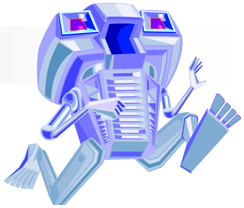
Localization: Absolute

- Proximity-To-Reference
  - Landmarks/Beacons
- Angle-To-Reference
  - Visual: manual triangulation from physical points
- Distance-From-Reference
  - Time of Flight
    - RF: GPS
    - Acoustic:
  - Signal Fading
    - Electromagnetic
    - Radio frequency
    - Acoustic

Localization: Relative



- If you know your speed and direction, you can calculate where you are relative to where you were (integrate).
- Speed and direction might, themselves, be absolute (compass, speedometer), or integrated (gyroscope, accelerometer)
- Relative measurements are usually more accurate in the short term -- but suffer from accumulated error in the long term
- Most robotics research seems to focus on this



Localization Methods

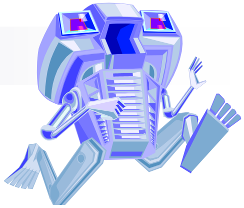
- Markov Localization:
  - Represent the robot’s belief by a probability distribution over possible positions and uses Bayes’ rule and convolution to update the belief whenever the robot senses or moves
- Monte-Carlo methods
- Kalman Filtering
- SLAM (simultaneous localization and mapping)

Localization and Map Building (5.1)

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

Challenges of Localization (5.2)

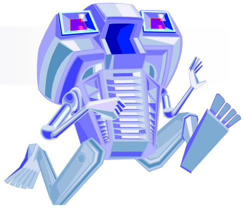
- Knowing the absolute position (e.g. GPS) is not sufficient
- *Localization* may also be required on a relative scale with respect to humans
- *Cognition* may require more than position, it may need to build an environmental model, *map*, to plan a path to a goal



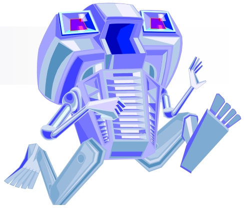
Sensor Noise (5.2)

- Perception (sensors) and motion control (effectors) play an integral role in localization
  - □ Sensor noise
  - □ Sensor aliasing
  - □ Effector noise
  - □ Odometric position estimation

Sensor Noise (5.2.1)

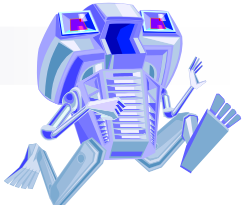


- ***Sensor noise*** induces a limitation on the consistency of sensor readings
- ***Sensor noise*** is mainly influenced by
  - □ environment (e.g. surface, illumination)
  - □ the measurement principle itself (e.g. interference between ultrasonic sensors)
- ***Sensor noise*** drastically reduces the useful information of sensor readings. The solution is:
  - □ to take multiple reading into account
  - □ employ temporal and/or multi-sensor fusion



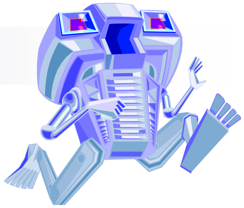
Sensor Aliasing (5.2.2)

- ***Sensor Aliasing*** describes the phenomena of the non-uniqueness of sensors readings. This is the norm in mobile robot sensors.
- Even with multiple sensors, there is a many-to-one mapping from environmental states to robot's perceptual inputs
- Therefore the amount of information perceived by the sensors is generally insufficient to identify the robot's position from a single reading
  - The robot's localization is usually based on a series of readings
  - This series may be sufficient information to recoever the robot's position over time



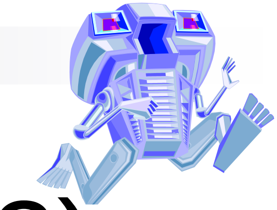
Effector Noise (5.2.3)

- Robot **effectors** are also noisy
  - Effectors produce uncertainty about future states
  - *Cognition* may be used to minimize uncertainty in motion
  - *Sensory feedback* can also be used to compensate for uncertainty
- Odometry and dead reckoning error
  - Position update is based on proprioceptive sensors
  - Robot is unable to estimate its own position over time using knowledge of its kinematics and dynamics
  - True source of error ia an incomplete model of the environment
  - *Odometry* uses wheel sensors only
  - *Dead reckoning* also uses heading sensors



Effector Noise (5.2.3)

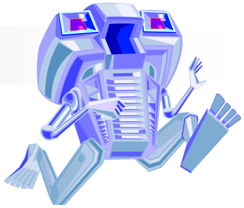
- The movement of the robot, sensed with wheel encoders and/or heading sensors is integrated to get position.
  - Pro
    - straight forward and easy
  - Con
    - errors are integrated and grow unbounded
- To correct robot pose
  - the position must be updated periodically by other localization mechanisms
  - Using additional heading sensors (e.g. gyroscope) might help to reduce the accumulated errors, but the main problems remain the same



Odometry Error sources (5.2.3)

- Major Error Sources:
  - *deterministic (systematic)*
    - can be eliminated by proper calibration of the system.
  - *non-deterministic (random)*
    - errors have to be described by error models and will always lead to uncertain position estimation
- Major Error Sources:
  - Misalignment of the wheels *[deterministic]*
  - Unequal wheel diameter *[deterministic]*
  - Limited resolution during integration (time increments, measurement resolution …) *[random]*
  - Variation in the contact point of the wheel *[random]*
  - Unequal floor contact (slipping, not planar …) *[random]*

Odometry: Classification of Integration Errors (5.2.3)



- *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 robots angular orientation

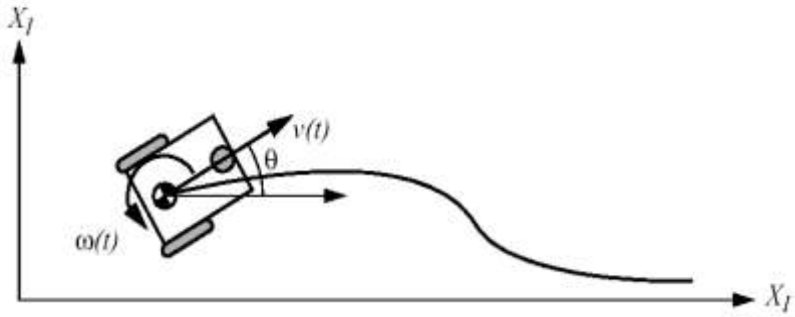
Over long periods of time, turn and drift errors far outweigh range errors because they are nonlinear!

  - □ 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 **d**$\mathbf{sin}\Delta\theta$, which can be quite large as the angular error $\Delta\theta$ grows.

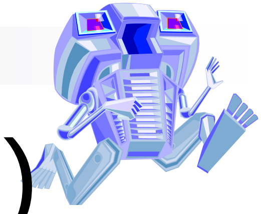
Error Model: Odometric Position Estimation (5.2.4)

- The pose of a robot is given by $[x\ y\ \theta]^T$
- The position can be estimated by starting from a known position and integrating the movement (sum the incremental travel distances)
- For a discrete system with a fixed sampling interval, Δt, the incremental travel distances are $(\Delta x, \Delta y, \Delta\theta)$
- Δs_r , Δs_l are the travelled distances for the right and left wheels
- b is the distance between the 2 wheels of the robot

$$\begin{aligned}
\Delta x &= \Delta s \cos(\theta + \Delta\theta / 2) \\
\Delta y &= \Delta s \sin(\theta + \Delta\theta / 2) \\
\Delta\theta &= \frac{\Delta s_r - \Delta s_l}{b} \\
\Delta s &= \frac{\Delta s_r + \Delta s_l}{2}
\end{aligned}$$



Odometric position update (5.2.4) Kinematics

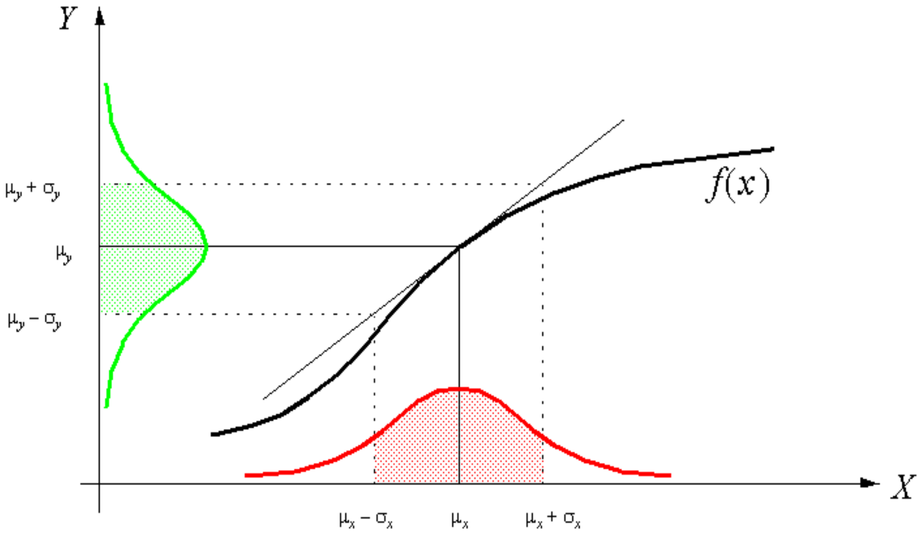


$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$$

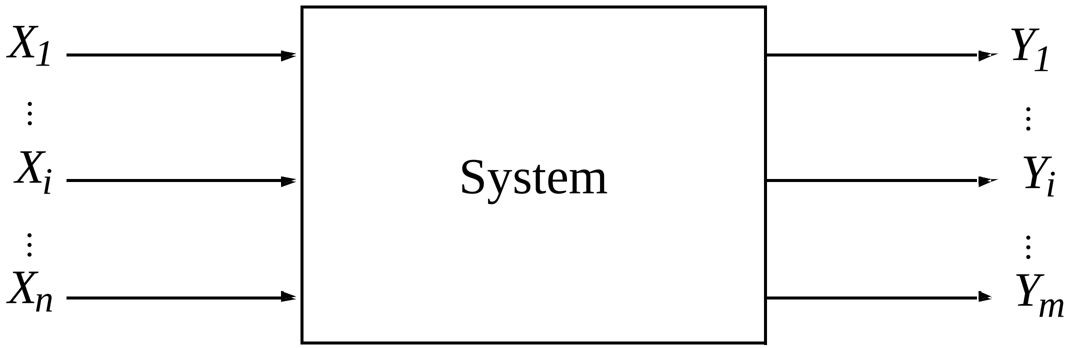
$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2}\cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2}\sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right)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

Error propagation (4.2.2)

- *Error propagation* is used when a series of measurements, all uncertain, can be fused to extract information about the environment
- If X_i are n input signals with known probability distribution and Y_i are m outputs
- What is the probability distribution of the output signals if the inputs are a function of f_i?



The Error Propagation Law (4.2.2)

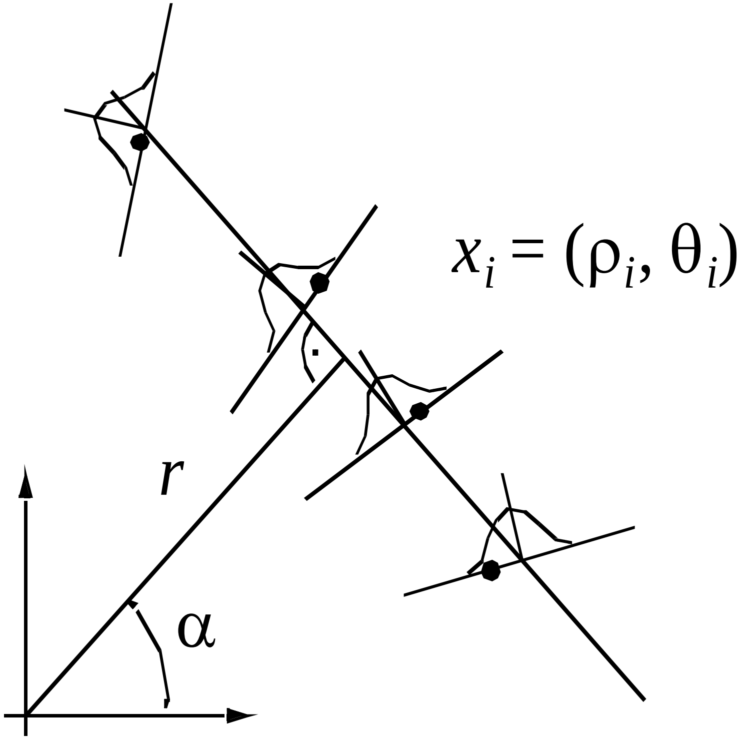


Error propagation in a multiple-input multi-output system with *n* inputs and *m* outputs.

$$Y_j = f_j\,(X_1 \ldots X_n)$$

The Error Propagation Law (4.2.2)

- Imagine extracting a line based on point measurements with uncertainties.
- The model parameters ρ_i (length of the perpendicular) and θ_i (its angle to the abscissa) describe a line uniquely
- The question:
  - □ What is the uncertainty of the extracted line knowing the uncertainties of the measurement points that contribute to it ?



The Error Propagation Law (4.2.2)

- It can be shown, that the output covariance matrix C_Y is given by the *error propagation law*, $C_Y = F_X C_X F_X^T$, where
  - □ C_X: covariance matrix representing the input uncertainties
  - □ C_Y: covariance matrix representing the propagated uncertainties for the outputs.
  - □ F_X: is the *Jacobian* matrix defined as:

$$F_X = \nabla f = \left[\nabla_X \cdot f(X)^T\right]^T = \begin{bmatrix} f_1 \\ : \\ f_m \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial X_1} & \cdots & \frac{\partial}{\partial X_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \cdots & \frac{\partial f_1}{\partial X_n} \\ : & \cdots & : \\ \frac{\partial f_m}{\partial X_1} & \cdots & \frac{\partial f_m}{\partial X_n} \end{bmatrix}$$

Error model for integrated position (p) (5.2.4)

- The covariance matrix for the error is given by
- k_r and k_l are the error constants for the nondeterministic parameters of the motor drive and wheel-floor interaction

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

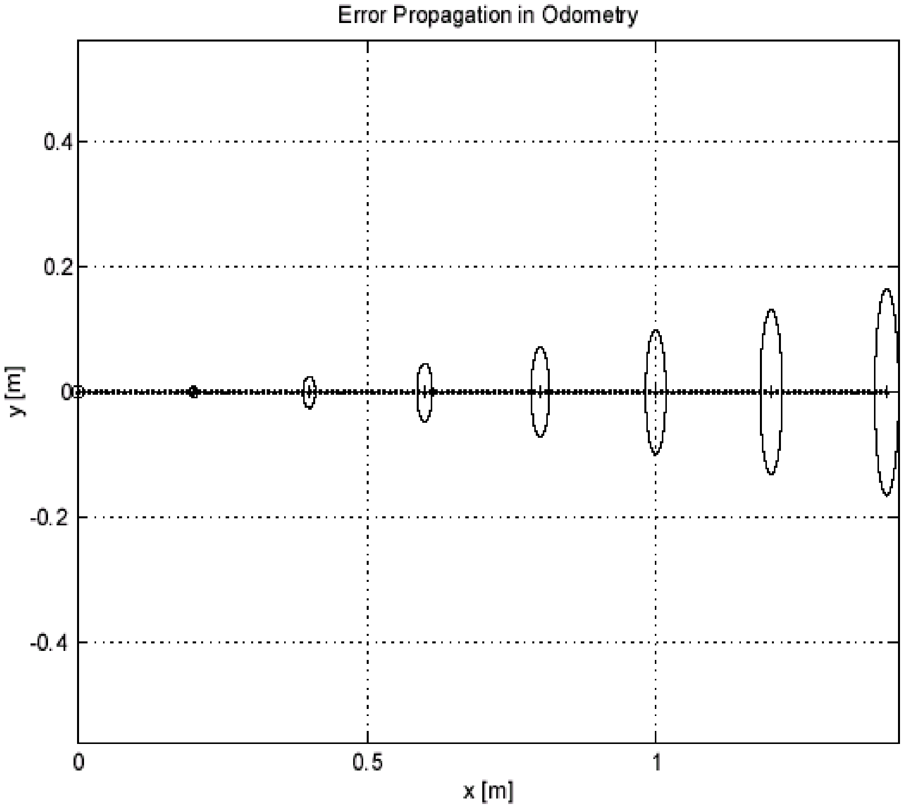
$$\Sigma_{p'} = \nabla_p f \cdot \Sigma_p \cdot \nabla_p f^T + \nabla_{\Delta_{rl}} f \cdot \Sigma_\Delta \cdot \nabla_{\Delta_{rl}} f^T$$

$$F_p = \nabla_p f = \nabla_p(f^T) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin(\theta + \Delta\theta/2) \\ 0 & 1 & \Delta s \cos(\theta + \Delta\theta/2) \\ 0 & 0 & 1 \end{bmatrix}$$

$$F_{\Delta_{rl}} = \begin{bmatrix} \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix}$$

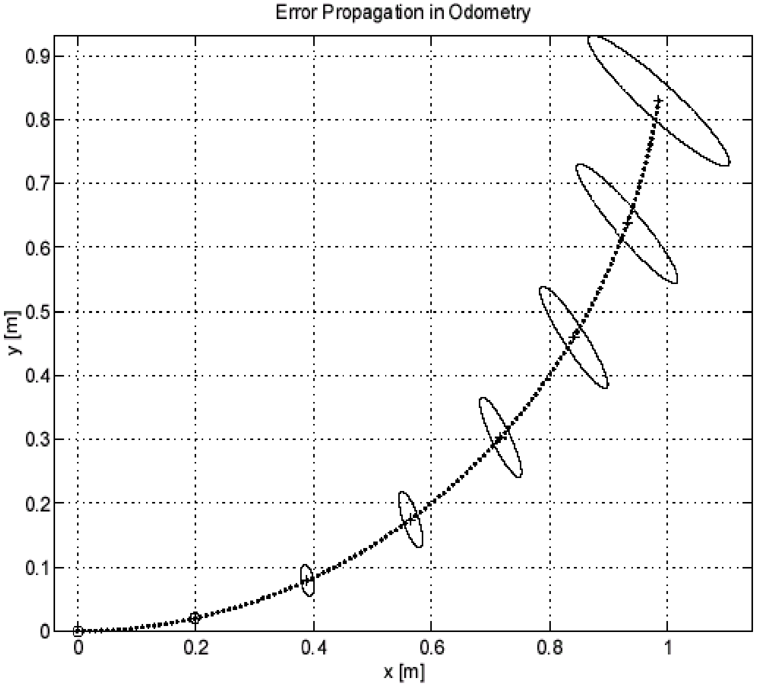
Odometry: Growth of Pose uncertainty for Straight Line Movement

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

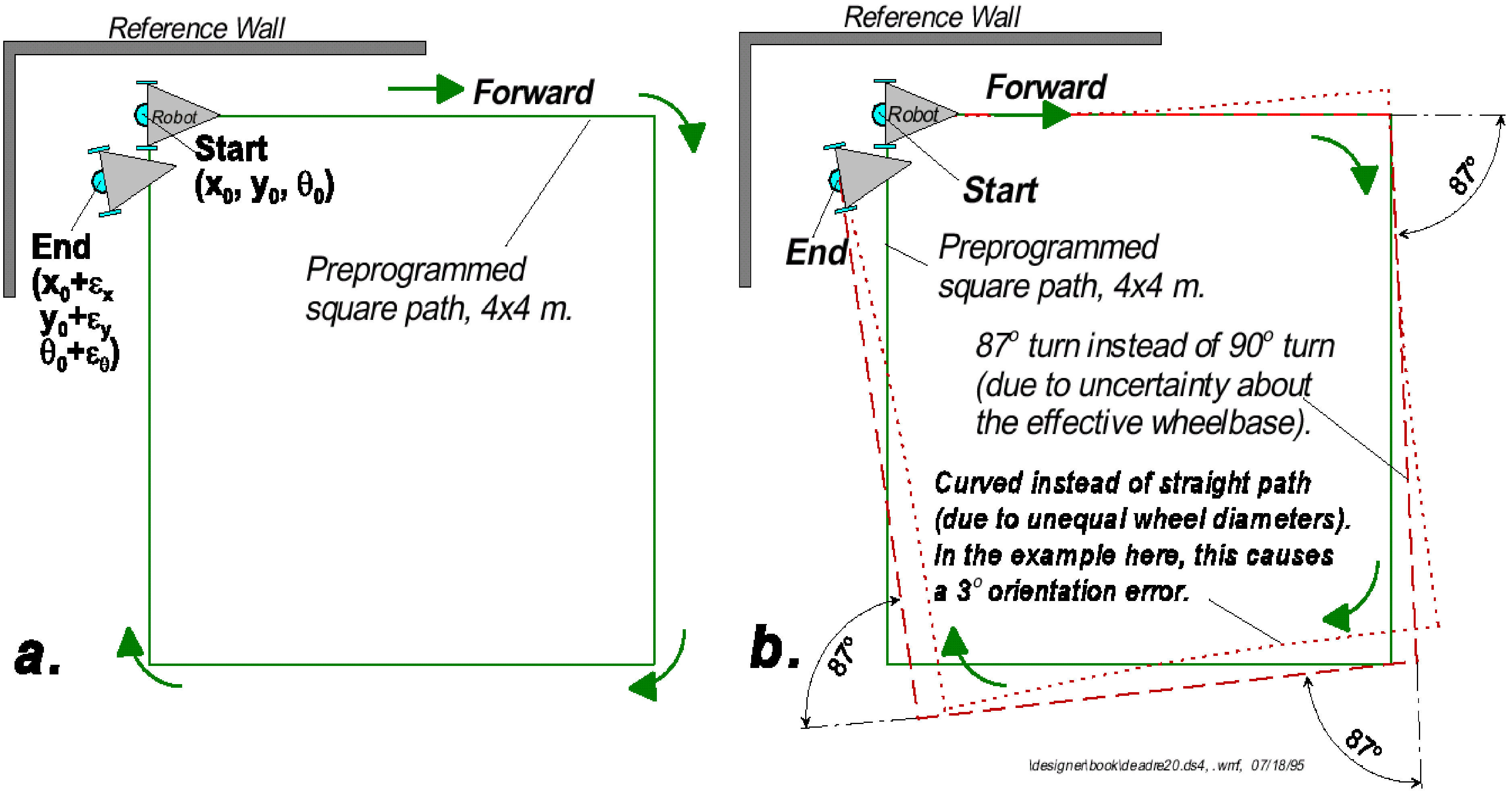


Odometry: Growth of Pose uncertainty for Movement on a Circle

Note: The uncertainty perpendicular to the movement grows faster than in the direction of movement. The main axis of the uncertainty ellipse does not remain perpendicular to the direction of movement!



Odometry: Calibration of Errors I



Preprogrammed square path, 4x4 m.

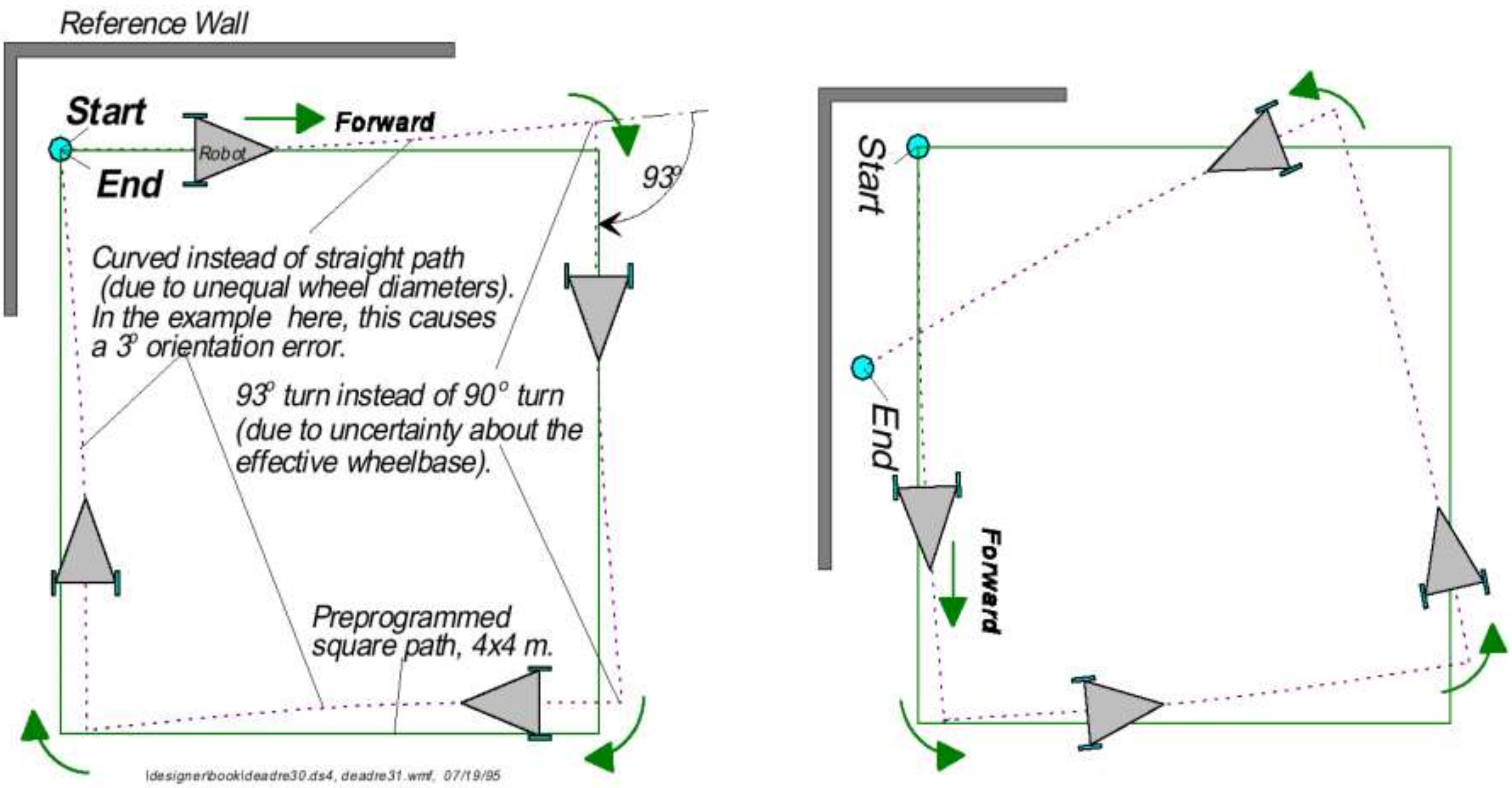
Preprogrammed square path, 4x4 m.

87° turn instead of 90° turn (due to uncertainty about the effective wheelbase).

Curved instead of straight path (due to unequal wheel diameters). In the example here, this causes a 3° orientation error.

\designer\book\deadre20.ds4, .wmf, 07/18/95

Odometry: Calibration of Errors II



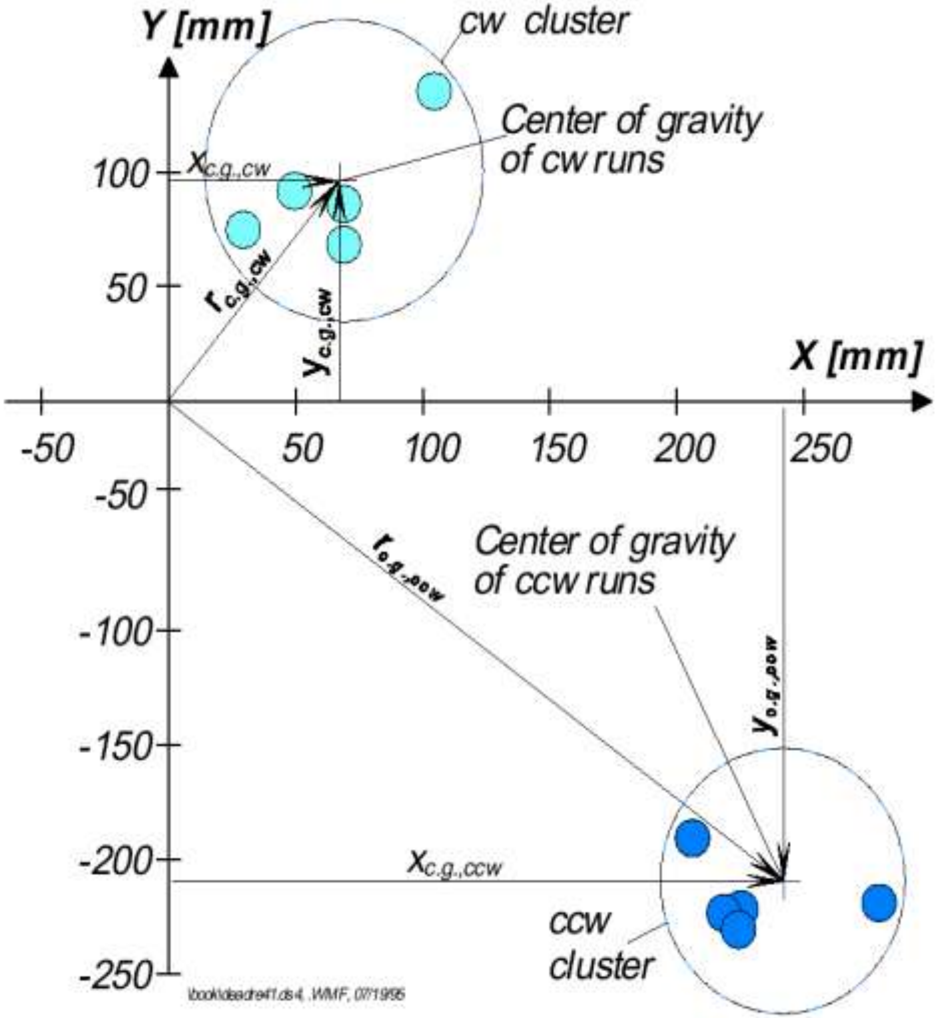
Curved instead of straight path (due to unequal wheel diameters). In the example here, this causes a 3° orientation error.

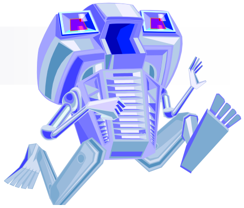
93° turn instead of 90° turn (due to uncertainty about the effective wheelbase).

Preprogrammed square path, 4x4 m.

\designer\book\deadre30.ds4, deadre31.wmf, 07/19/95

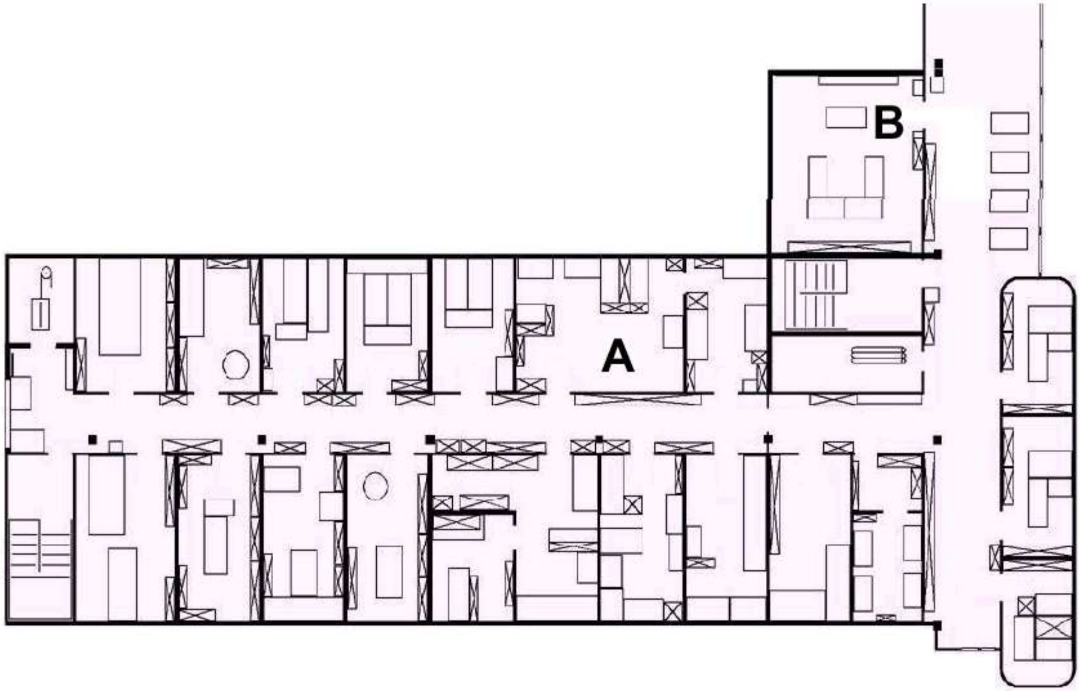
Odometry: Calibration of Errors III





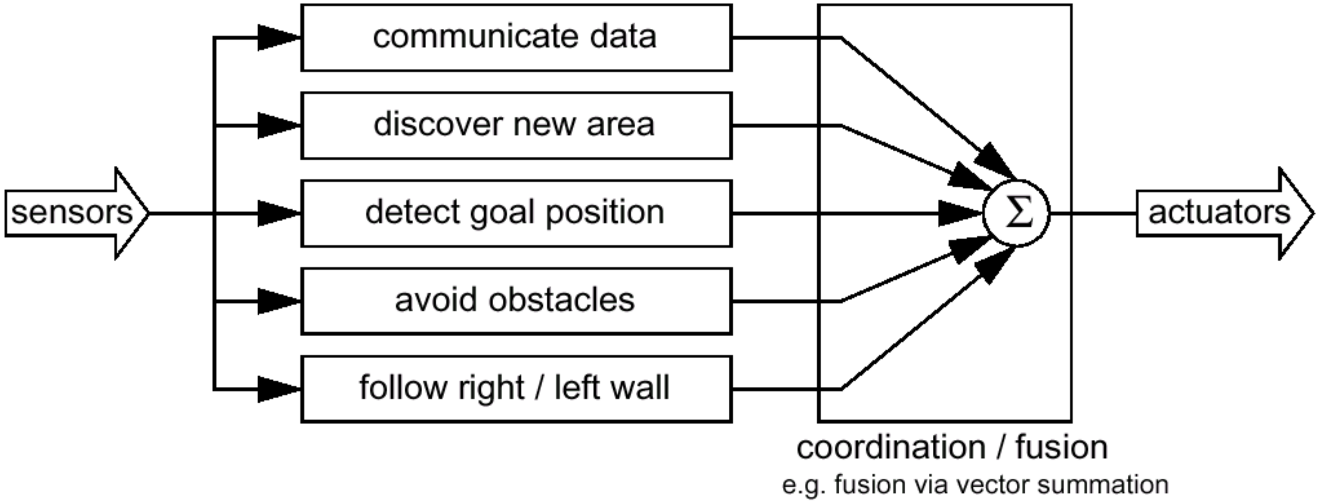
To localize or not? (5.3)

- How to navigate between A and B,
  - □ Use *localization* with respect to a map to navigate to the goal B
  - □ Use *behavior-based navigation* without hitting obstacles
    - Follow walls with obstacle avoidance
    - Detect the goal location



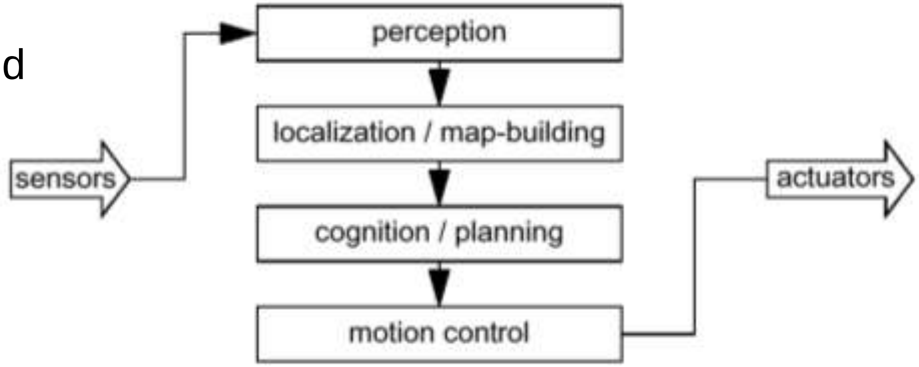
Behavior Based Navigation (5.3)

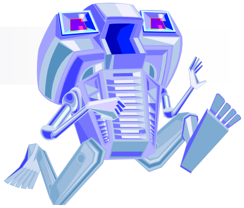
- Advantage:
  - Implemented quickly for a single environment
- Disadvantages:
  - Does not scale directly to different or larger environments
  - Navigation code is location-specific
  - Behaviors must be carefully designed
  - May have several active behaviors at once



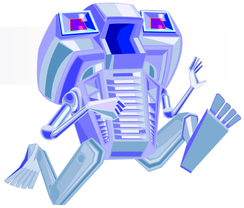
Model Based Navigation (5.3)

- The *map-based approach* includes both *localization* and *cognition*
- Advantages:
  - □ Make's the robot's belief about position transparent to the human operator
  - □ The existence of the map represents a medium of communication between the human and robot
  - □ The map can be used by humans as well as the robots
  - □ Can map and navigate a variety of environments
- Disadvantages:
  - □ Requires more up-front development effort to create a navigating mobile robot
  - □ An internal representation rather than the real-world itself is being constructed and trusted by the robot
  - □ If the model diverges from reality, the robot's behavior will be undesirable even if the sensor values are transiently incorrect





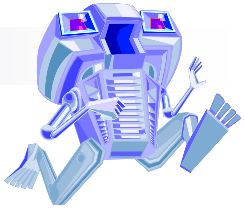
Belief Representation (5.4)



Representation

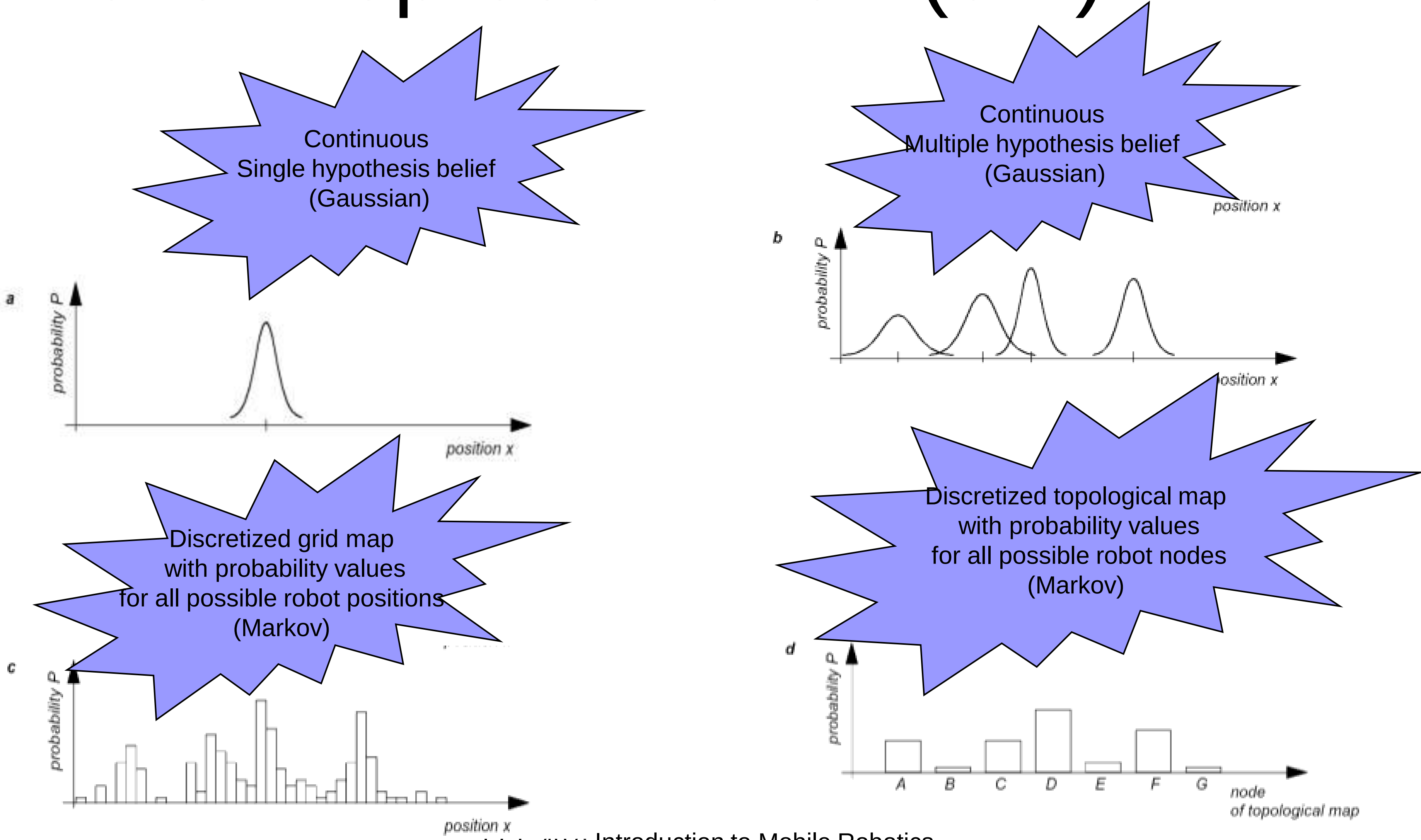
- The robot internal state that stores information about the world is called a ***representation***
  - Environment: maps
  - Objects: people, doors, other robots
  - Tasks: what needs to be done and in what order
  - Self: goals, sensors, plans, proprioception
- *Representations* or *internal models* influence the complexity of a robot’s “brain”

Belief Representation (5.4)



- The fundamental issue that differentiates map-based localization systems is *representation*
  - *Map representation*
    - Robot’s model of the environment, or a map
    - At what level of fidelity does the map represent the environment?
  - *Belief representation*
    - Robot’s belief of its position on the map
    - Does the robot identify a single unique position?
    - Does the robot describe its position in terms of a set of possible positions?
    - How are multiple positions ranked

Belief Representation (5.4)

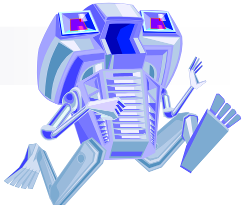


Continuous
Single hypothesis belief
(Gaussian)

Continuous
Multiple hypothesis belief
(Gaussian)

Discretized grid map
with probability values
for all possible robot positions
(Markov)

Discretized topological map
with probability values
for all possible robot nodes
(Markov)

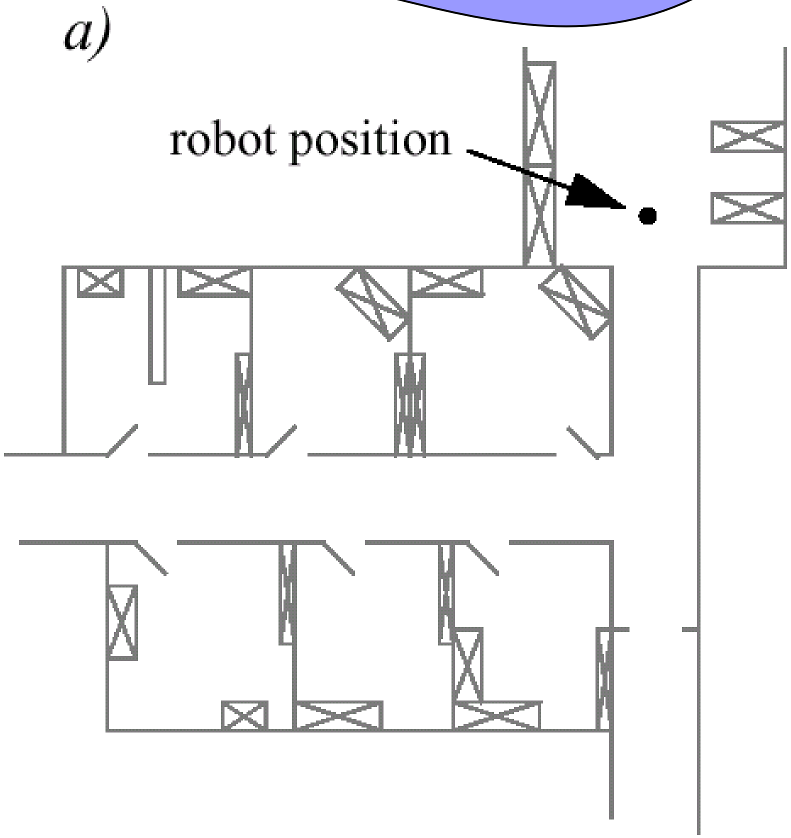


Belief Representation: Characteristics (5.4)

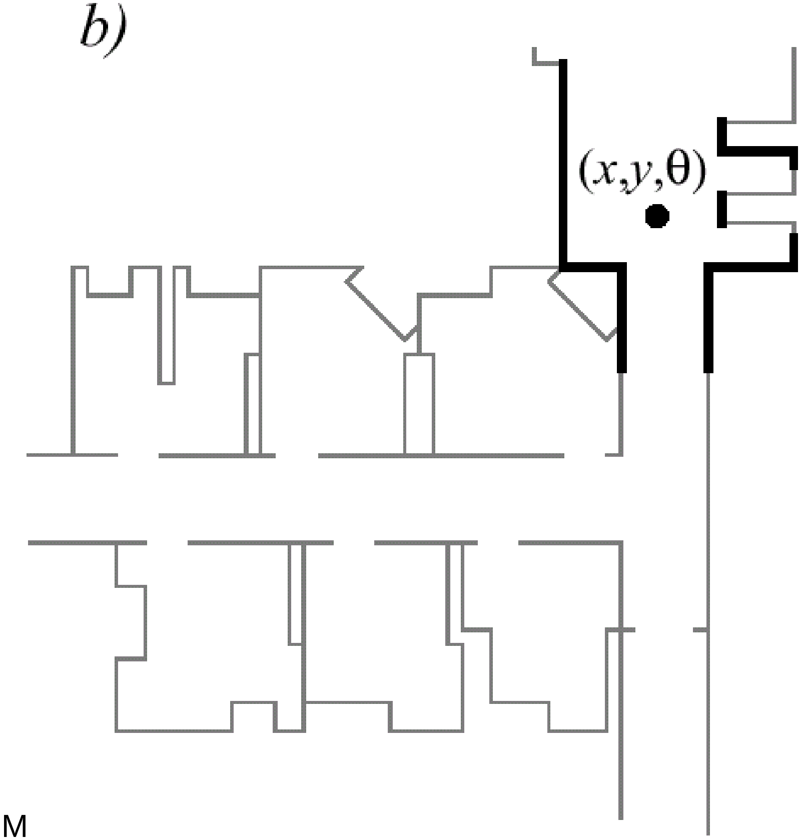
- Continuous
  - Precision bound by sensor data
  - Typically single hypothesis pose estimate
  - Lost when diverging (for single hypothesis)
  - Compact representation and typically reasonable in processing power.
- Discrete
  - Precision bound by resolution of discretization
  - Typically multiple 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 (5.4.1)

Real-world map with walls, doors, and furniture

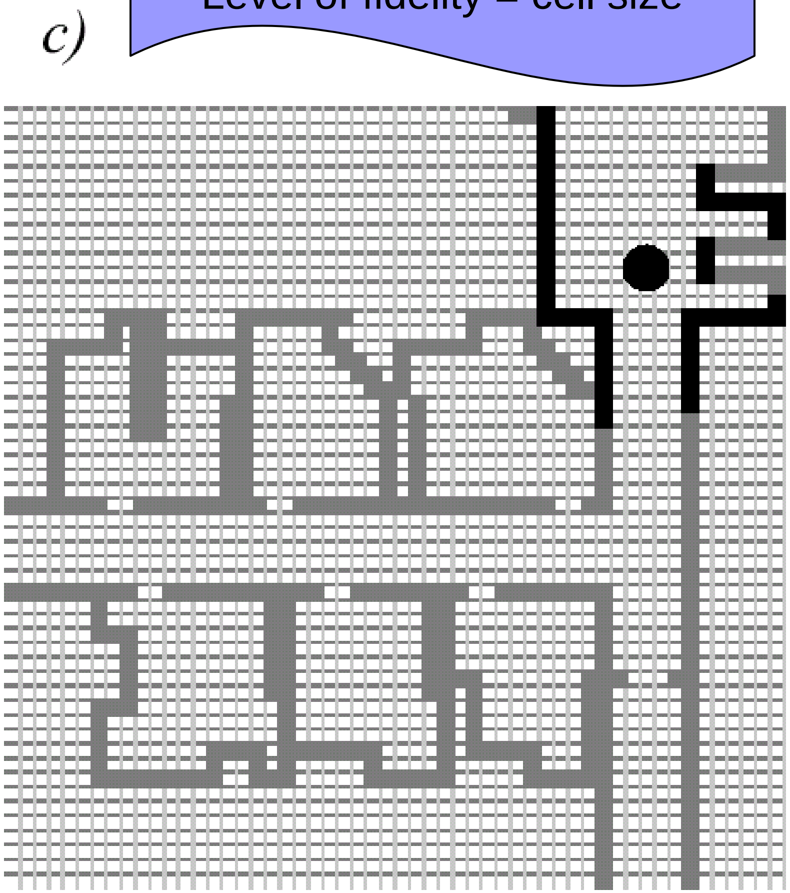


Continuous 2D geometric line-based map

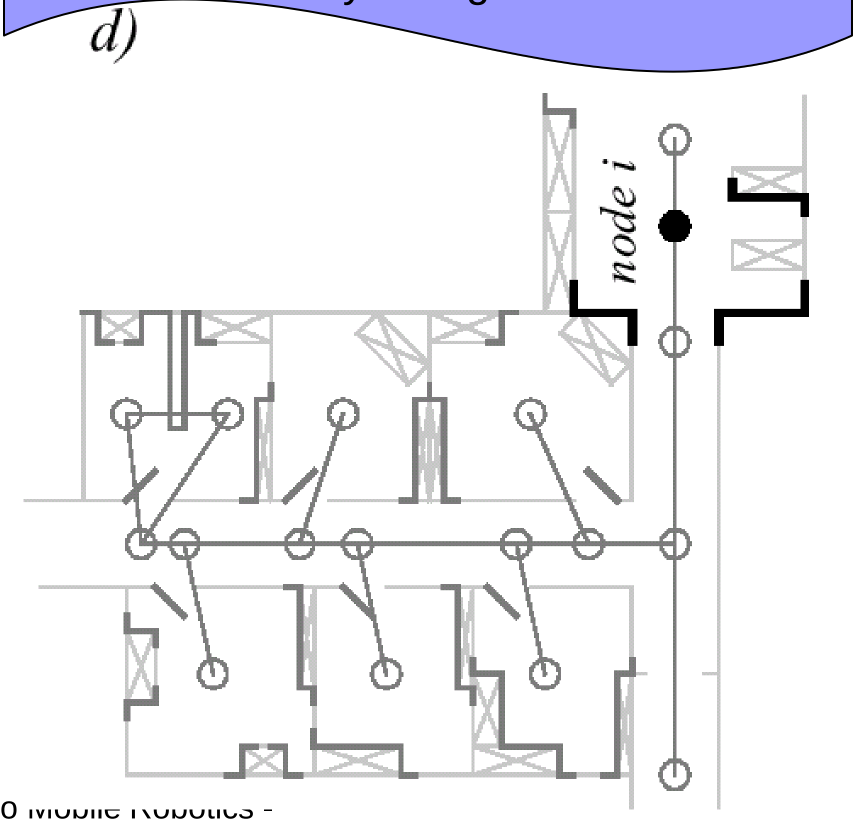


Single-hypothesis Belief (5.4.1)

Discrete, tessellated map
Level of fidelity = cell size

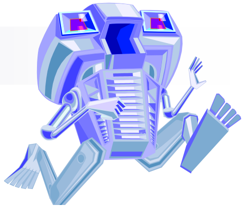


The map is not geometric, but
abstract and topological
Identify a single node



Single-hypothesis belief (5.4.1)

- Advantages:
  - Given a unique belief, there is no position ambiguity
  - Facilitates decision-making at robot’s cognitive level (e.g. path planning)
- Disadvantages:
  - Robot motion induces uncertainty due to effector and sensor noise
  - Forcing the position update to always generate a single hypothesis of position is challenging

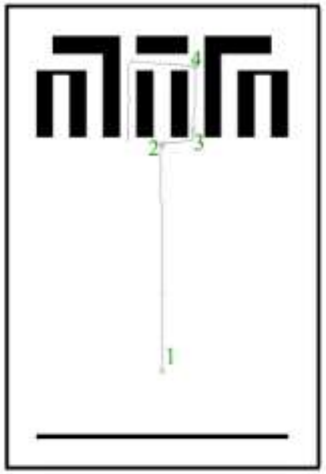


Multiple-hypothesis belief (5.4.2)

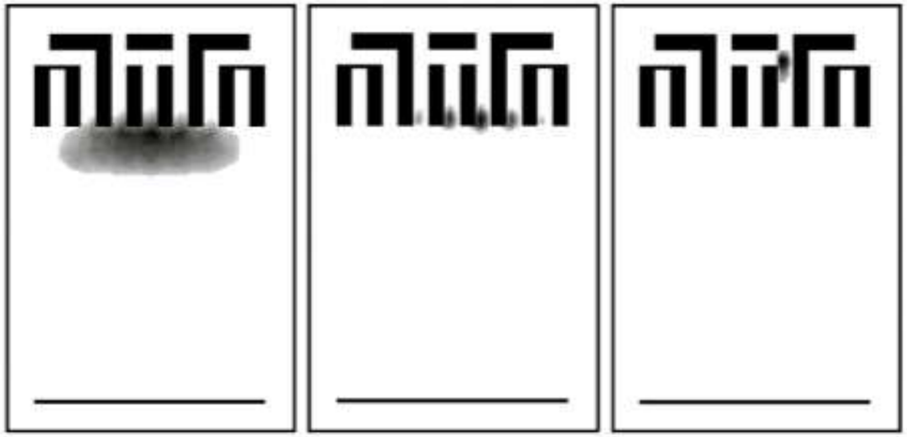
- The robot tracks an infinite set of possible positions
- This set can be described geometrically as a convex polygon positioned on a 2D map (continuous or discrete)
- In this method, the possible robot positions are not ranked
- To rank the positions requires a model of the beliefs as a mathematical distribution (Gaussian probability density function)

Multi Hypothesis Grid-based Representation (5.4.2)

- Discrete markers for each possible position
- Each position is noted along with a confidence or probability parameter
- Thousands of possible positions for a highly tessellated map



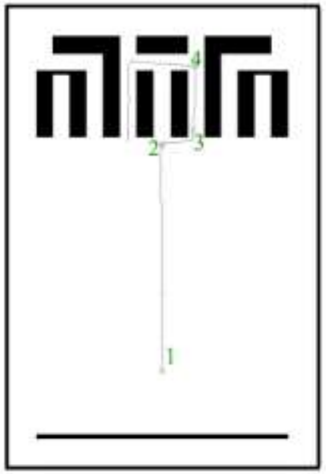
Path of the robot



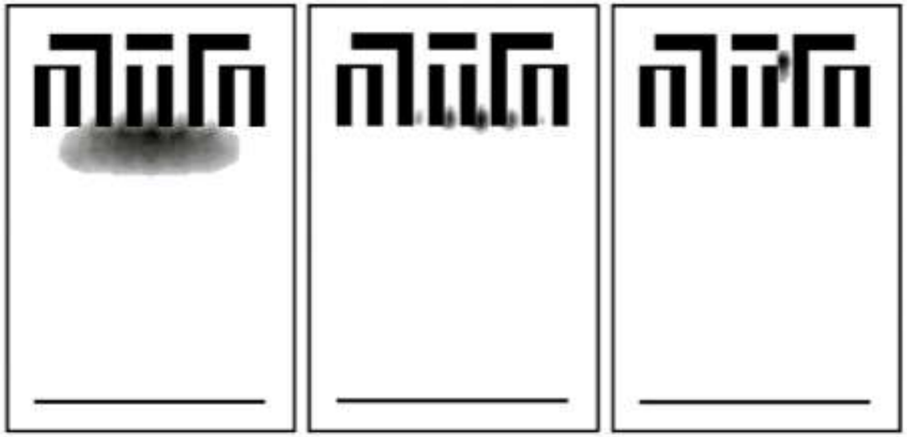
Belief states at positions 2, 3 and 4

Multi Hypothesis Grid-based Representation (5.4.2)

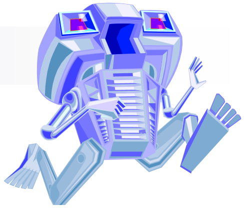
- Advantages:
  - Robot maintains a sense of position while explicitly annotating its own uncertainty about the position
  - Partial information from sensors and effectors can update the belief
  - Robot is able to explicitly measure its own degree of uncertainty regarding position
- Disadvantages:
  - In decision making, how does the robot decide what to do next?
  - Each position must have an associated probability
  - Computationally expensive



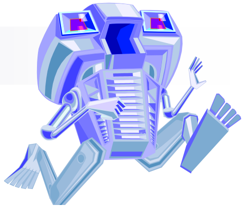
Path of the robot



Belief states at positions 2, 3 and 4



Map Representation (5.5)

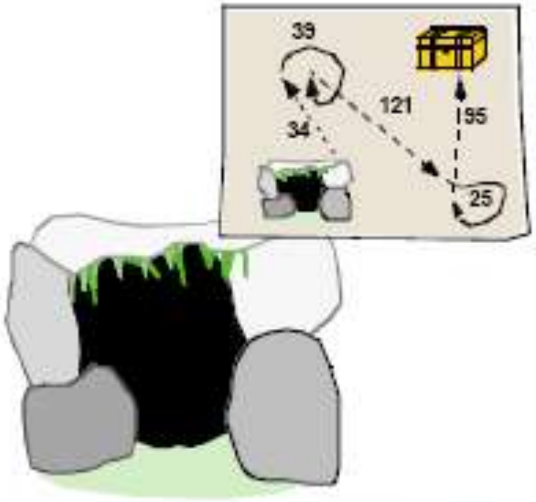


Map Representation (5.5)

- The problem of representing the robot’s environment is the dual of representing the possible robot position(s)
- Three fundamental relationships:
  - *Map precision vs. application precision for robot to achieve goals*
  - *Feature type and map precision vs. sensor precision and data types*
  - *Map or computational complexity vs. reasoning (i.e. mapping. localization, navigation)*

Environment Representation and Modeling: Techniques

- Odometry



How to find a treasure

  - *not applicable*

- Modified Environments

Landing at night

  - *expensive, inflexible*

- Feature-based Navigation

Elevator door

Corridor crossing

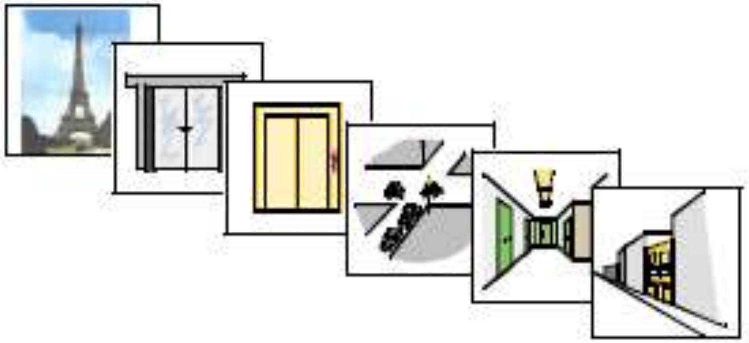
Eiffel Tower

Entrance

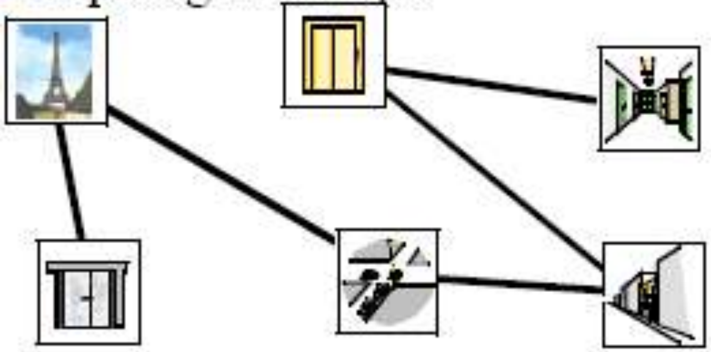
  - *still a challenge for artificial systems*

Environment Representation: Map Categories

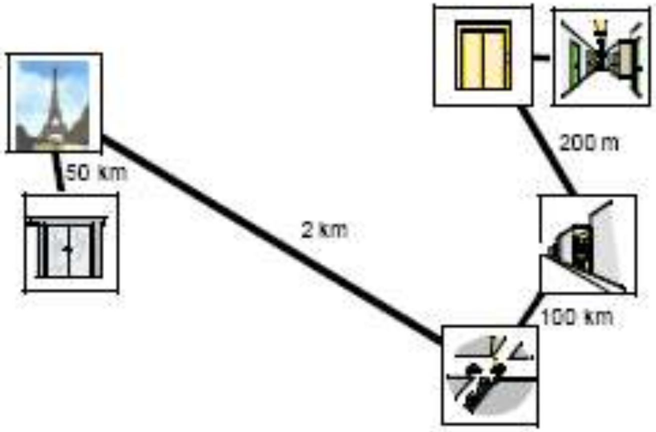
- Recognizable Locations



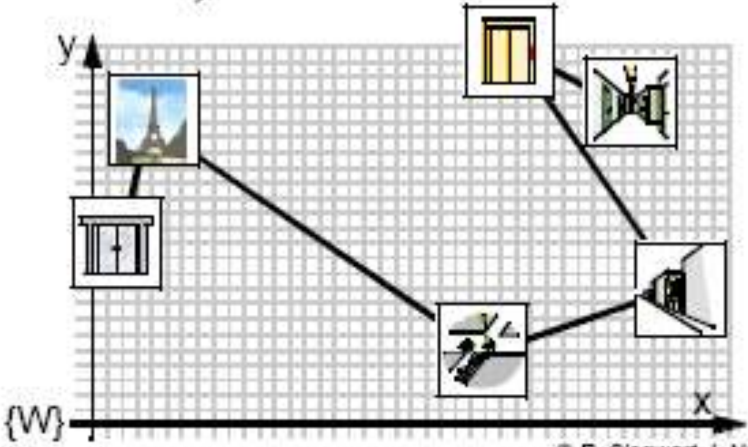
- Topological Maps



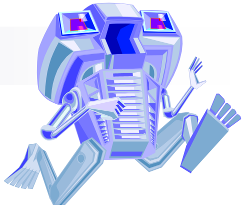
- Metric Topological Maps



- Fully Metric Maps (continuos or discrete)



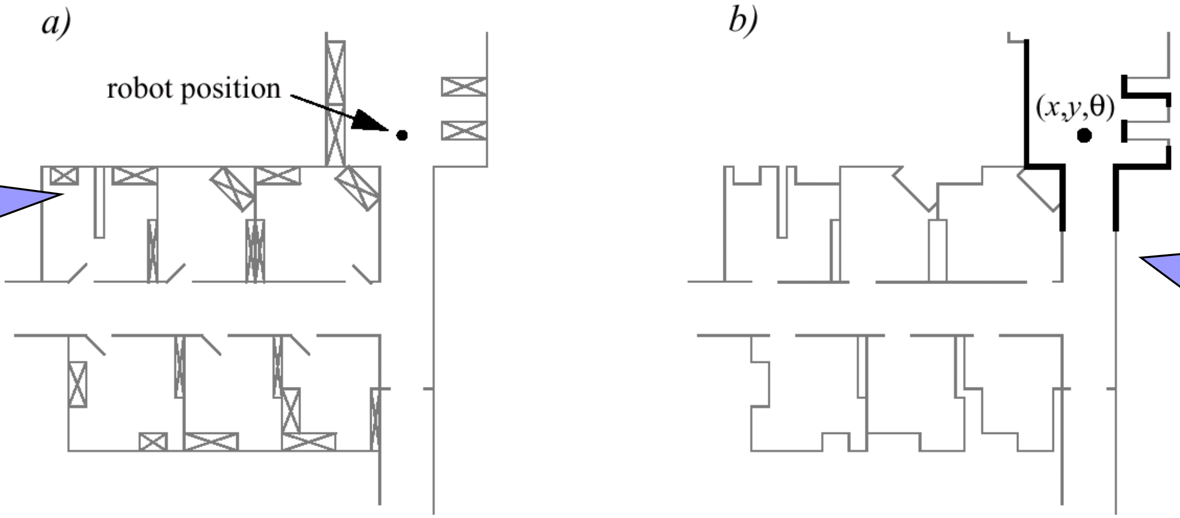
© R. Siegwart, I. Nourbakh



Environment Models

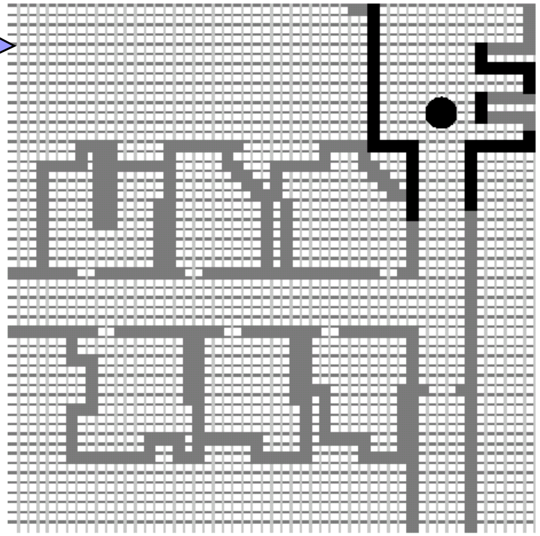
- Continuous versus Discrete Data
  - Position in x, y, θ
  - Metric or topological grid
- Raw Data versus Features
  - Raw data represents information that is perceived by a sensor
  - A ***feature*** (or natural landmark) is an environmental structure which is static, and always perceptible with the current sensory system

Environment Representation

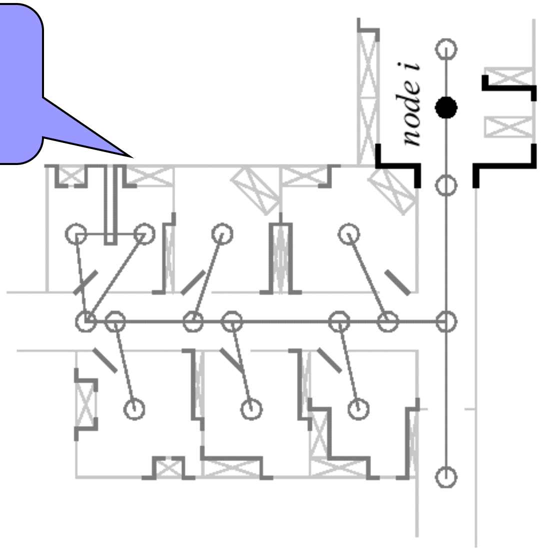


world map

continuous metric

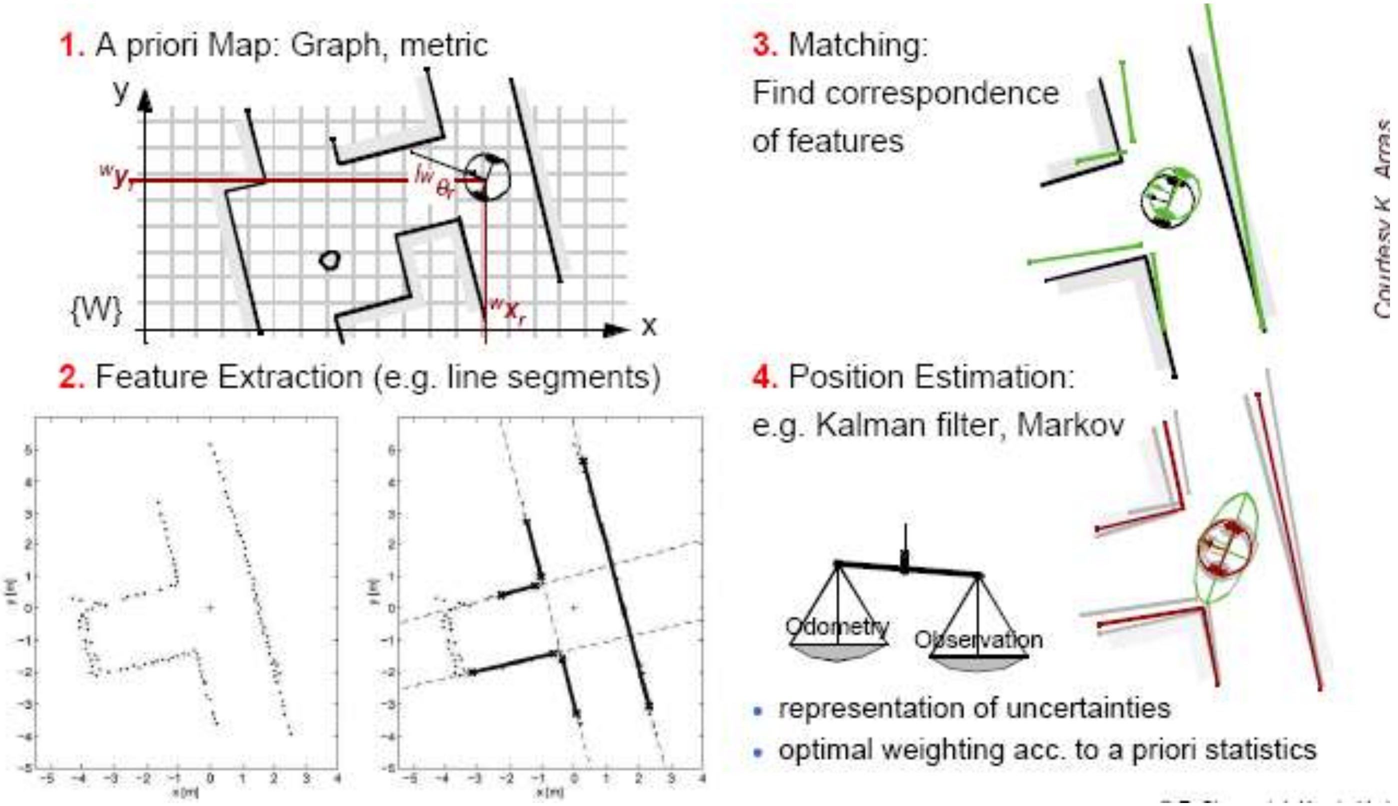


metric grid



topological metric

Methods for Localization: Quantitative Metric Approach



1. A priori Map: Graph, metric

2. Feature Extraction (e.g. line segments)

3. Matching: Find correspondence of features

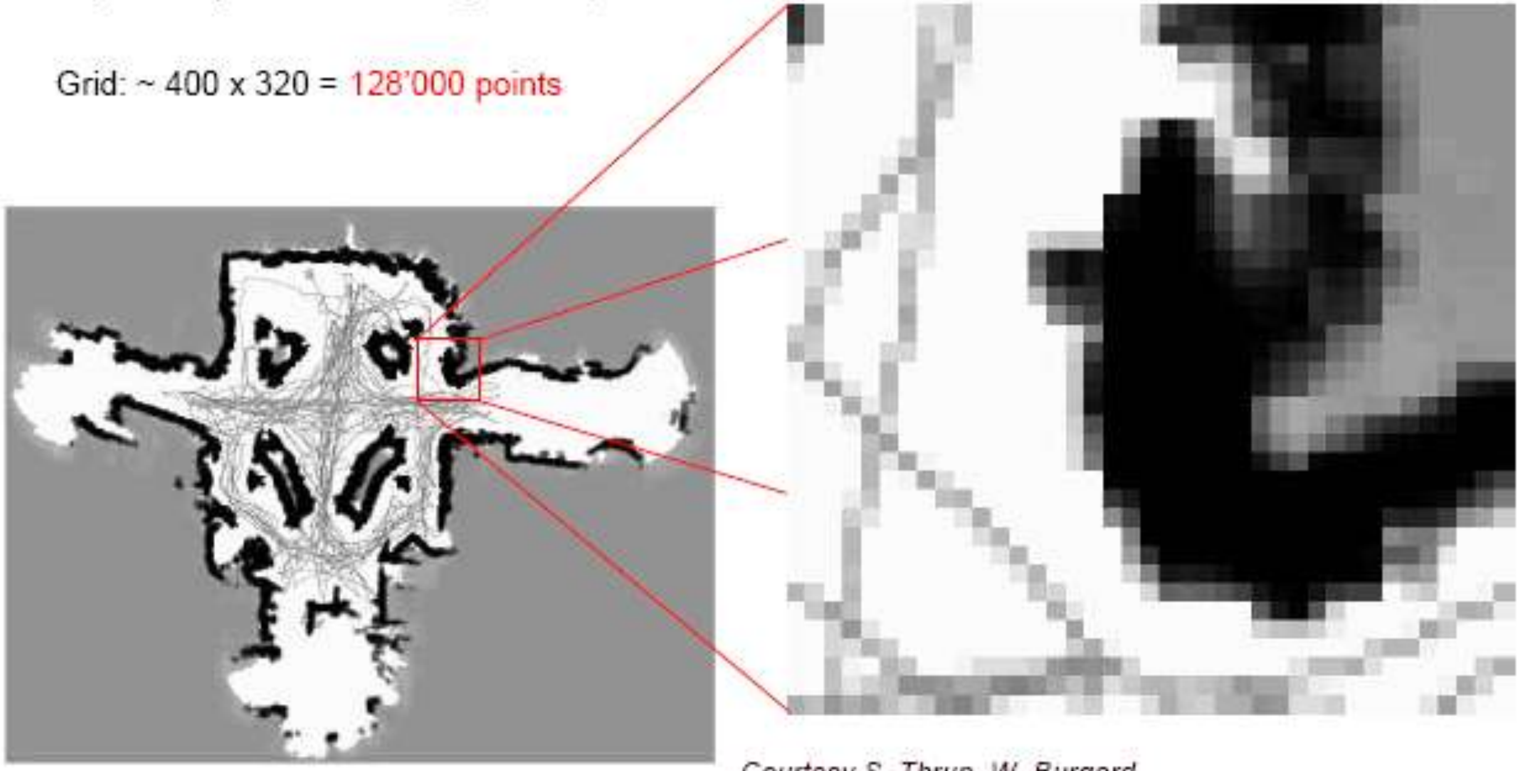
4. Position Estimation: e.g. Kalman filter, Markov

- representation of uncertainties
- optimal weighting acc. to a priori statistics

Courtesy K. Arras

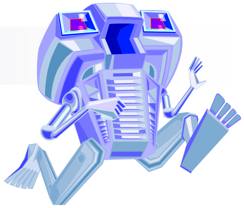
Methods for Localization: Grid-Based Metric Approach

- Grid Map of the Smithsonian's National Museum of American History in Washington DC. (Courtesy of Wolfram Burger et al.)



Grid: ~ 400 x 320 = 128'000 points

Courtesy S. Thrun, W. Burgard



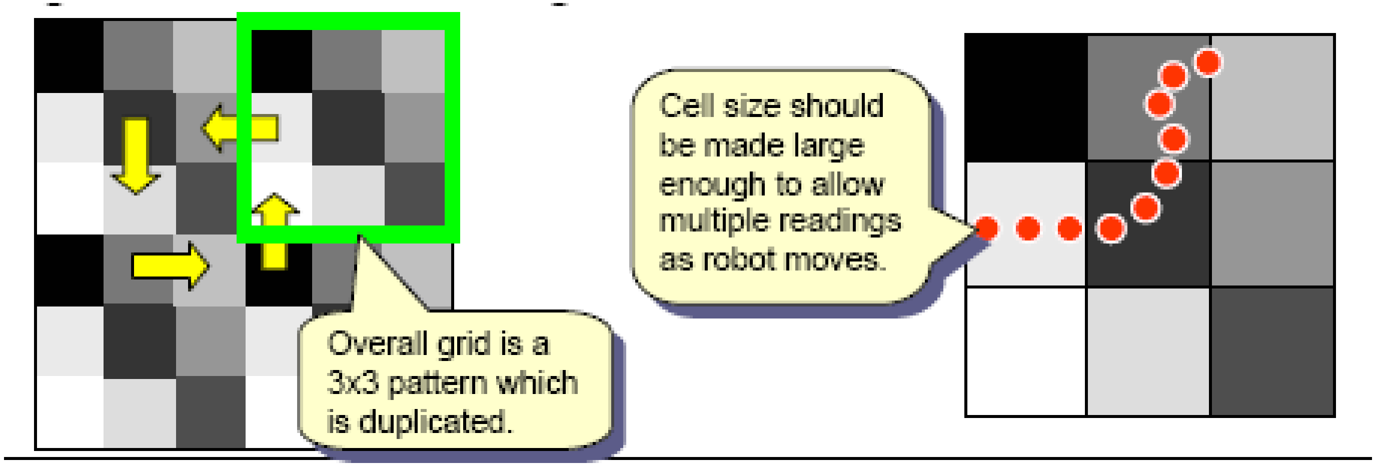
Grid-Based Localization

Grid tracking

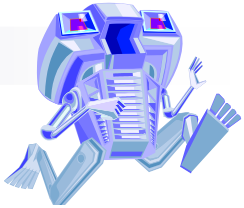
- Another strategy for position estimation is to do grid tracking
- Place a grid on the floor with clearly identifiable cells
- The robot senses change from one cell to another

Grid design

- A robot is equipped with a light sensor
- Grid must be designed to distinguish changes from one cell to another
- Must maximize the contrast between adjacent cells
- Grid cells must be larger when the robot moves faster



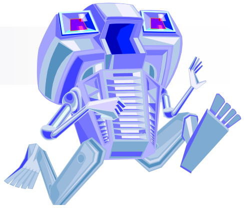
Overall grid is a 3x3 pattern which is duplicated.
Cell size should be made large enough to allow multiple readings as robot moves.



Grid tracking

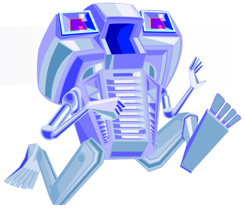
- Advantages
  - Can re-confirm location after short distances, eliminate errors within 1 cell range
  - Simple to implement
- Disadvantages
  - Cell size limits accuracy
  - Requires many sensor readings and large cells for truly reliable estimations
  - Requires modification of the environment
  - Result depends on print quality and sensor calibration

Active Beacons



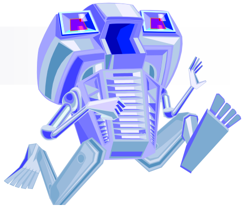
Active Beacons

- An *active beacon* is a stationary device that transmits and/or receives signals
- Multiple beacons must be installed for proper position estimation
- The robot estimates position and orientation by determining distance and angle to each of these beacons



Active Beacons

- Beacon systems are based on triangulation
- Two types of triangulation techniques
  - □ *Lateration*
    - Determine robot’s position based on distance from beacons
    - 2D requires 3 non-collinear points
  - □ *Angulation*
    - Determine robot’s position and angle based on angle to beacons
    - 2D requires 2 angles and 1 known distance

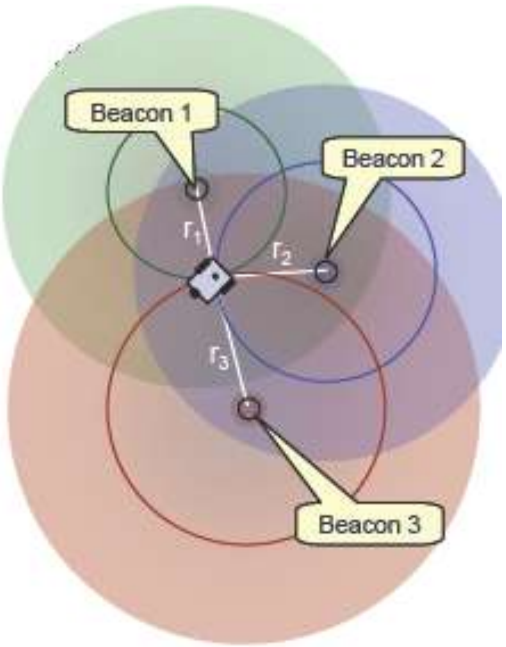


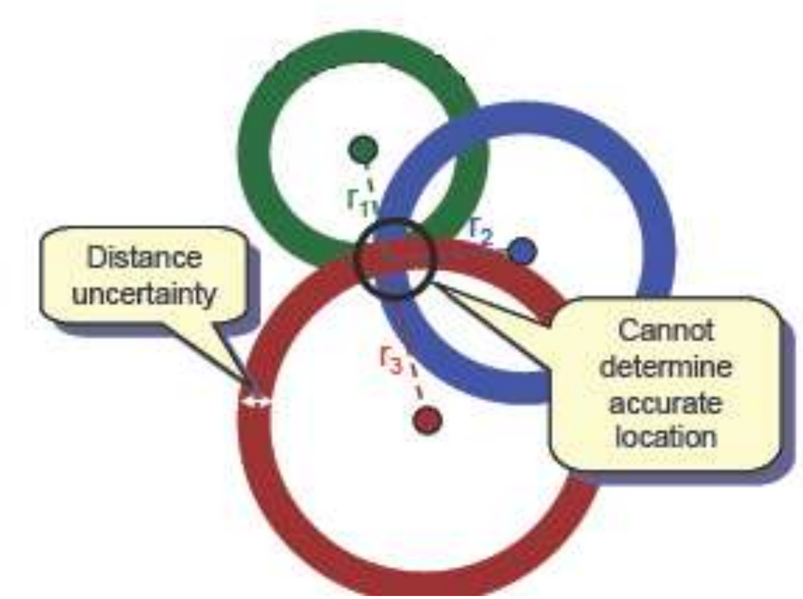
Triangulation - Lateration

- 3 or more beacons emit a signal, robot obtains distance to each beacon
  - □ Direct Measurement
    - Robot physically moves or sends probe
    - Simple, but difficult to implement
  - □ Time of Flight
    - Measure time it takes to travel to known point at specific velocity
    - Usually measure the difference in transmission and arrival time of an emitted signal
  - □ Attenuation
    - Measure signal strength which decreases as distance from emission source increases

Triangulation - Lateration

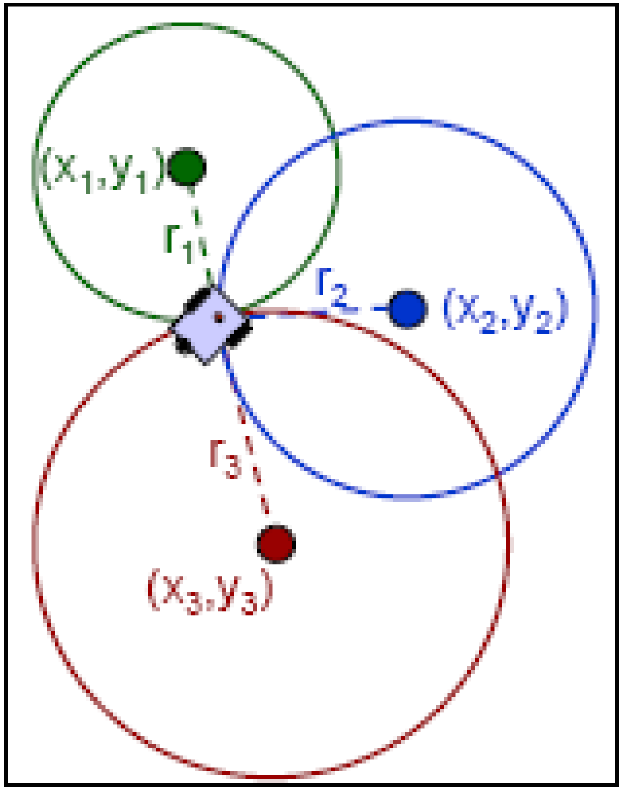
- Location is the intersection of 3 circles using distances as radii
- Accuracy depends on precision of distances



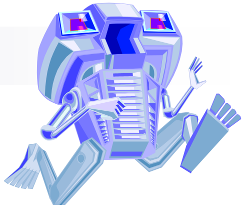


Triangulation - Lateration

- The following 3 equations represent the 3-beacon scenario
- The intersection point is the intersection of the 3 circles
- $(x - x_1)^2 + (y-y_1)^2 = r_1^2$
- $(x - x_2)^2 + (y-y_2)^2 = r_2^2$
- $(x - x_3)^2 + (y-y_3)^2 = r_3^2$



$$[x = 2y(y_1 - y_2) - x_1^2 + x_2^2 - y_1^2 + y_2^2 + r_1^2 + r_2^2]/(-2(x_1 - x_2))$$

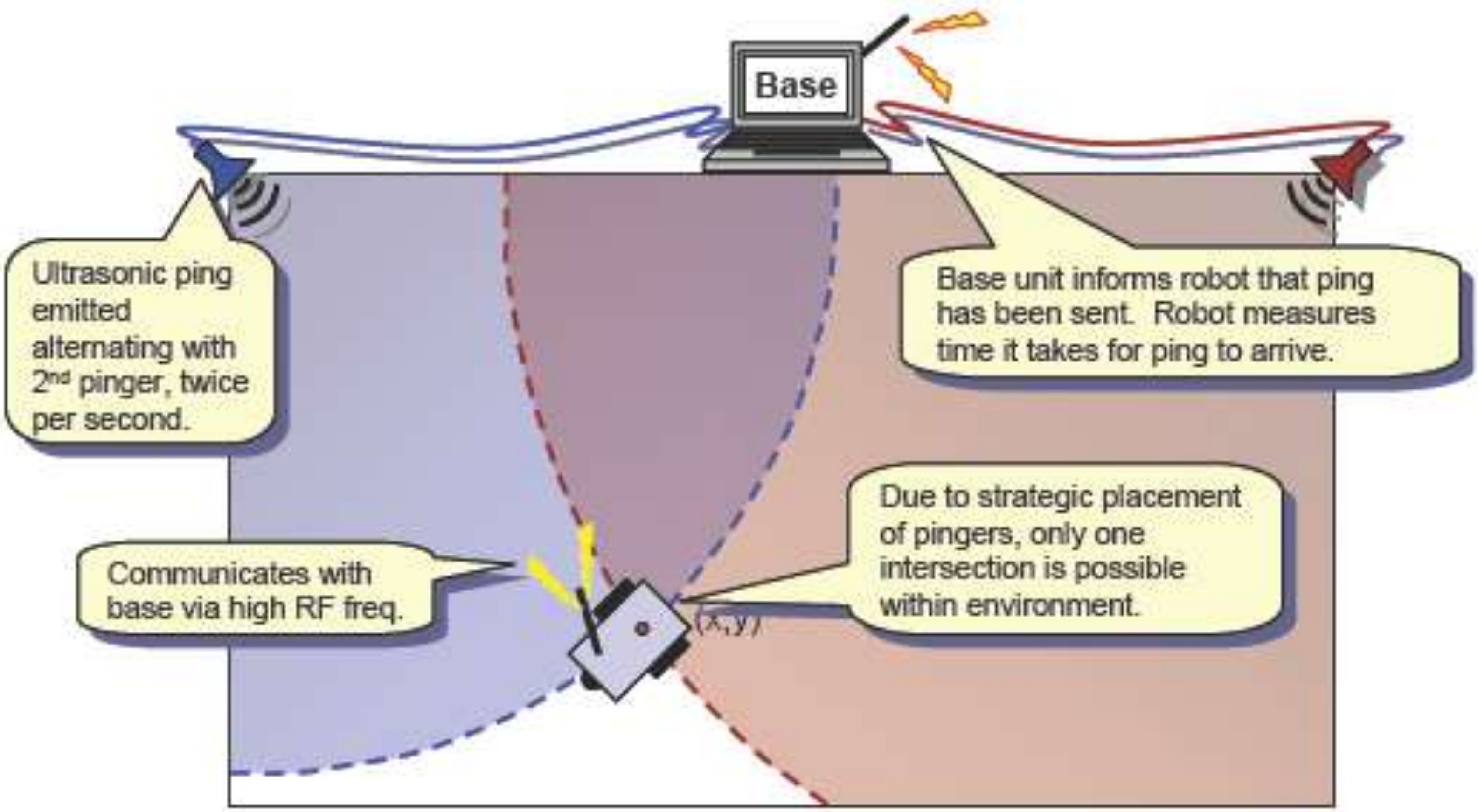


Triangulation - Lateration

- Must be careful not to have a divide by zero condition when placing beacons
- Some beacons are fixed and yield fixed pre-computed constants

$$(x, y) = C_o + C_1r_1^2 + C_2r_2^2 + C_3r_3^2$$

Triangulation – Lateration using ultrasonic pings



Ultrasonic ping emitted alternating with 2nd pinger, twice per second.

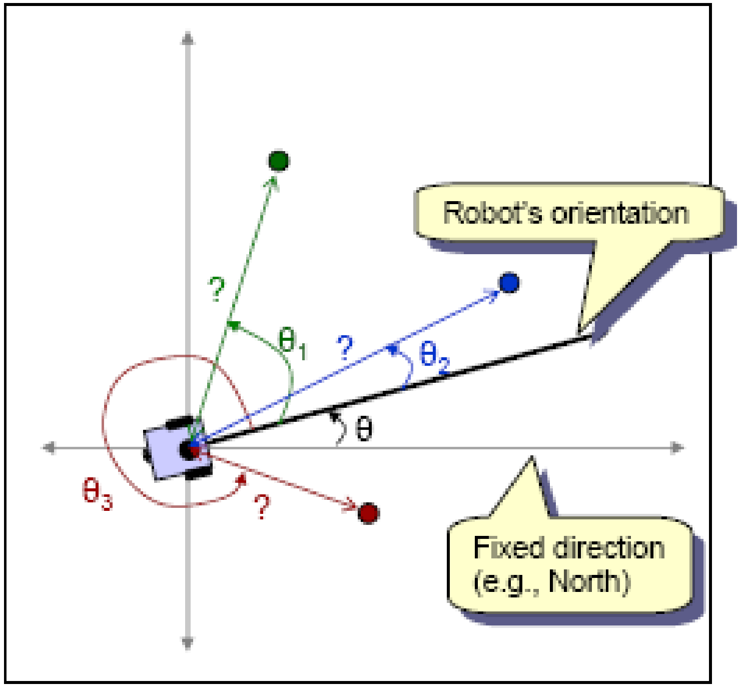
Base unit informs robot that ping has been sent. Robot measures time it takes for ping to arrive.

Communicates with base via high RF freq.

Due to strategic placement of pingers, only one intersection is possible within environment.

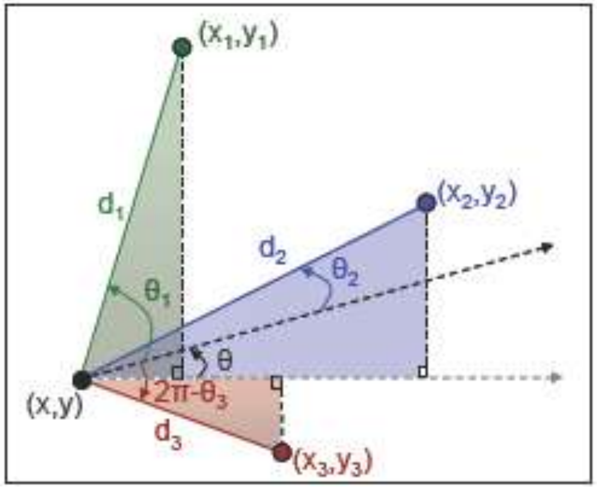
Triangulation - Angulation

- *Angulation* makes use of angles to beacons as opposed to distances to them
- Angles to beacons measured via rotation of receiver or transmitter on robot
- Assumes all beacons are visible

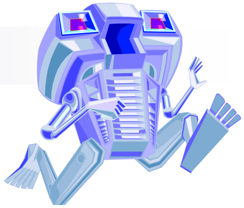


Triangulation - Angulation

- $d_1 \sin(\theta + \theta_1) = y_1 - y$
- $d_1 \cos(\theta + \theta_1) = x_1 - x$
- Repeat for all triangles and setting the pairs equal yields



- $(y_1 - y)\cos(\theta + \theta_1) = (x_1 - x) \sin (\theta + \theta_1)$
- Yields 3 equations and 3 unknowns



Triangulation - Angulation

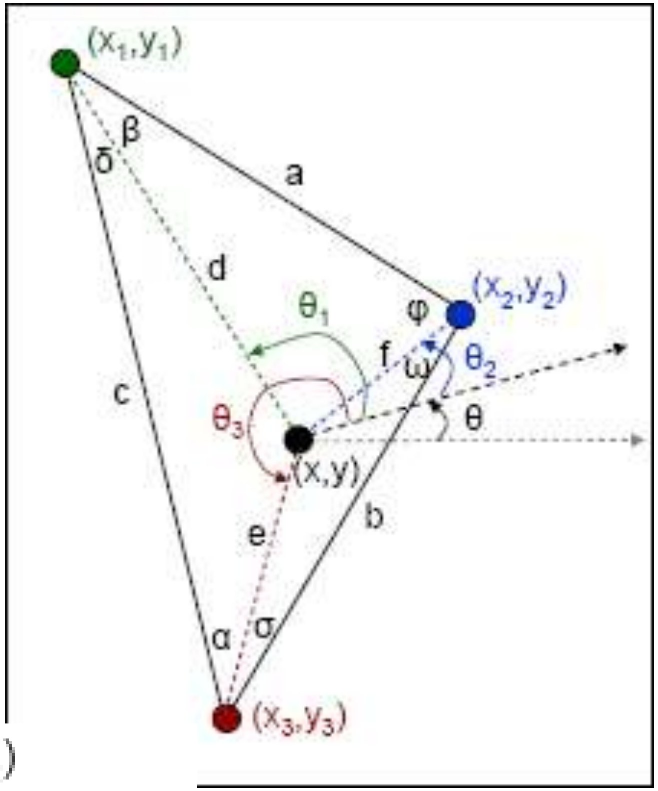
$$x = [y_3 - y_1 + x_1\tan(\theta+\theta_1) - x_3\tan(\theta+\theta_3)] / (\tan(\theta+\theta_1) - \tan(\theta+\theta_3))$$

$$x = [y_3 - y_2 + x_2\tan(\theta+\theta_2) - x_3\tan(\theta+\theta_3)] / (\tan(\theta+\theta_2) - \tan(\theta+\theta_3))$$

- Just need to solve for θ
- In many real situations, the value of θ is known
  - From a digital compass
  - Estimated from odometry

Triangulation - Angulation

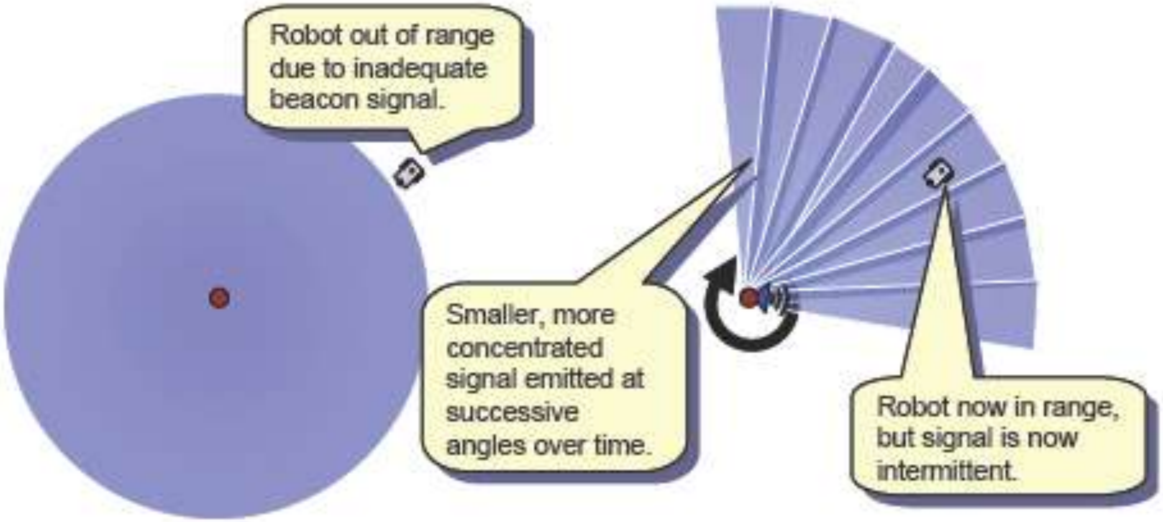
- *Geometric triangulation* applications assume that the robot will be within the area defined by 3 or more beacons
- Make use of the formula
- $a^2 = b^2 + c^2 - 2bc\cos\alpha$
- $(\sin\alpha)/a = (\sin\beta)/b = (\sin\gamma)/c$



$$c\cdot\sin(\pi - \theta_3 + \theta_1 - \delta) / \sin(\theta_3 - \theta_1) = a\cdot\sin(\varphi) / \sin(\theta_1 - \theta_2)$$
$$b\cdot\sin(\sigma) / \sin(2\pi - (\theta_3 - \theta_2)) = a\cdot\sin(\pi - \theta_1 + \theta_2 - \varphi) / \sin(\theta_1 - \theta_2)$$
$$c\cdot\sin(\delta) / \sin(\theta_3 - \theta_1) = b\cdot\sin(-\pi + \theta_3 - \theta_2 - \sigma) / \sin(2\pi - (\theta_3 - \theta_2))$$

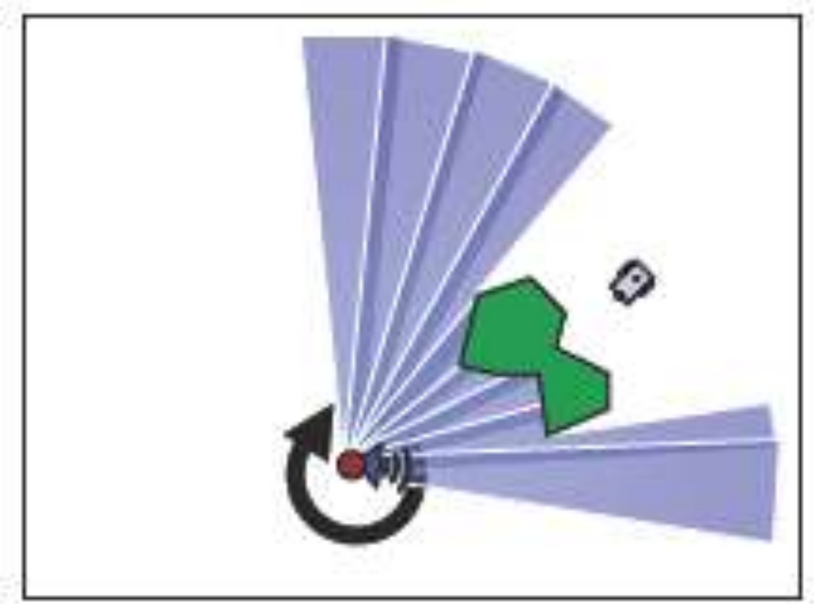
Beacons

- Beacons must be extremely powerful to ensure omni-directional transmission over large distances
- Compromise is to focus the beam and rotate it via some pattern


Robot out of range due to inadequate beacon signal.
Smaller, more concentrated signal emitted at successive angles over time.
Robot now in range, but signal is now intermittent.

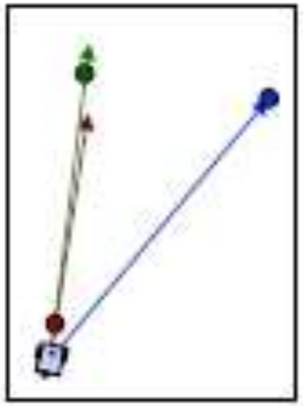
Beacons

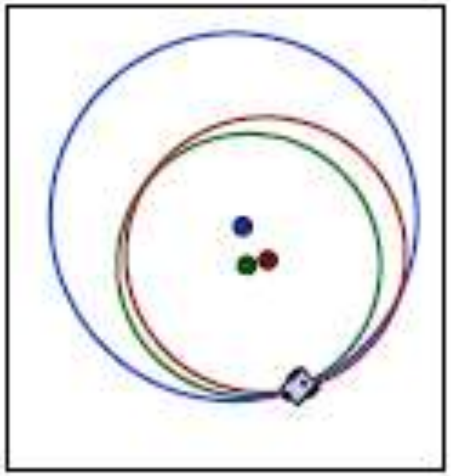
- Beacons may not be visible in some areas due to obstructions from obstacles
- The robot may need to rely on odometry until a reading is available again

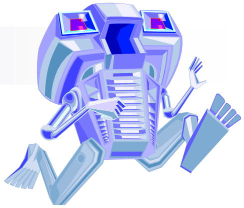


Issues

- Triangulation is sensitive to smaller angular errors
  - □ When observed angles are small
  - □ When measured angles are indistinguishable
  - □ When the robot is far from beacons, it can be difficult to determine position accurately







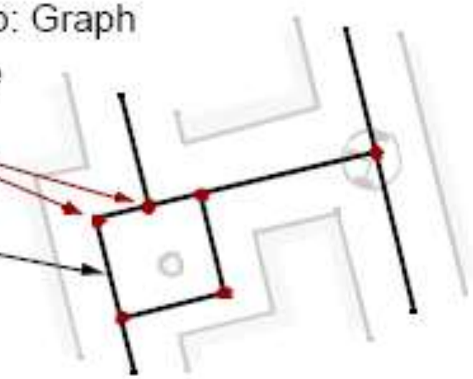
Topological Localization

Methods for Localization: Quantitative Topological Approach

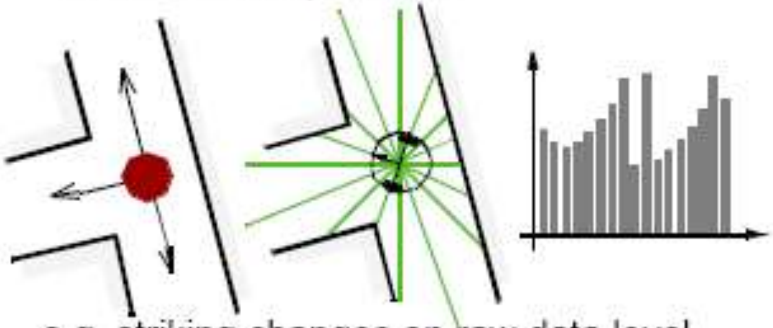
1. A priori Map: Graph

locally **unique** points

edges



2. Method for determining the **local uniqueness**

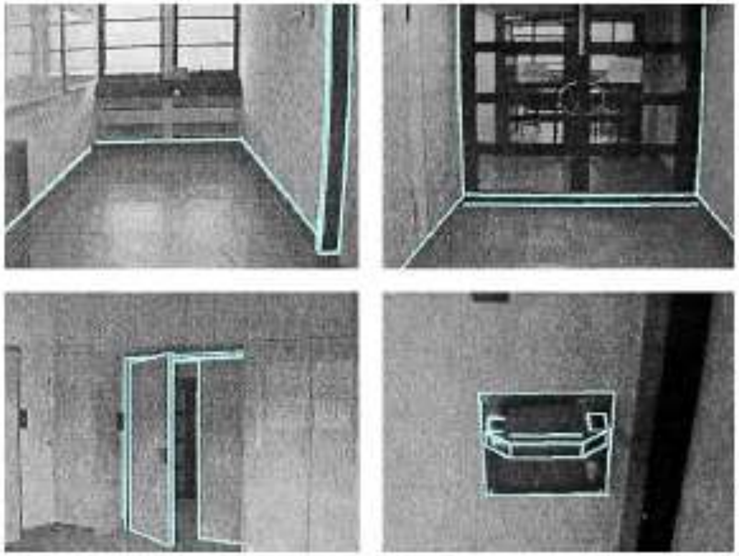


e.g. striking changes on raw data level or highly distinctive features

3. Library of **driving behaviors**

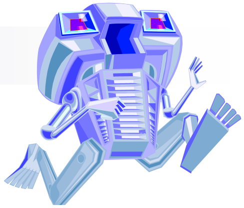
e.g. wall or midline following, blind step, enter door, application specific behaviors

Example: Video-based navigation with natural landmarks



Courtesy of [Lanser et al. 1996]

Map Building

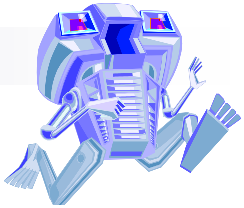


Techniques:

- Manual
  - Drawn by hand
  - Static/predictable environment
  - Costly
- Automatically
  - Robot learns environment
  - Dynamically/unpredictable changing
  - Different look due to different perception

Requirements:

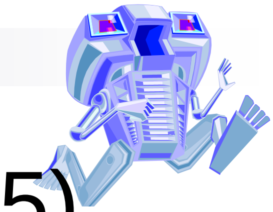
- Incorporates newly sensed information into the existing world model
- Contains information to estimate the robot’s position
- Provides Information to do path planning and navigation tasks



Map Building: Measure of Quality

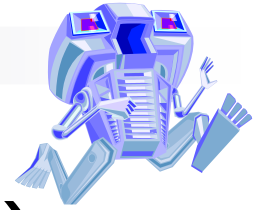
- Topological correctness
- Metrical correctness

Most environments are a mixture of ***predictable*** and ***unpredictabl***e features (hybrid approach)



Representation of the Environment (5.5)

- Environment Representation
  - Continuous Metric → x,y,θ
  - Discrete Metric → metric grid
  - Discrete Topological → 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

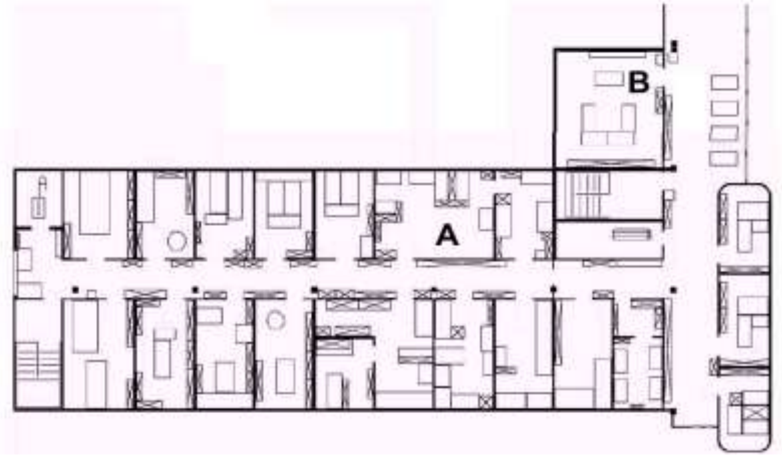


Continuous representation (5.5.1)

- A continuous-valued map is one method for exact *decomposition* of the environment
- Continuous maps are only in 2D representations as further dimensionality can result in computational explosion
- Combine the exactness of continuous representation with the compactness of *closed-world assumption*
- The representation will specify all environmental objects in the map

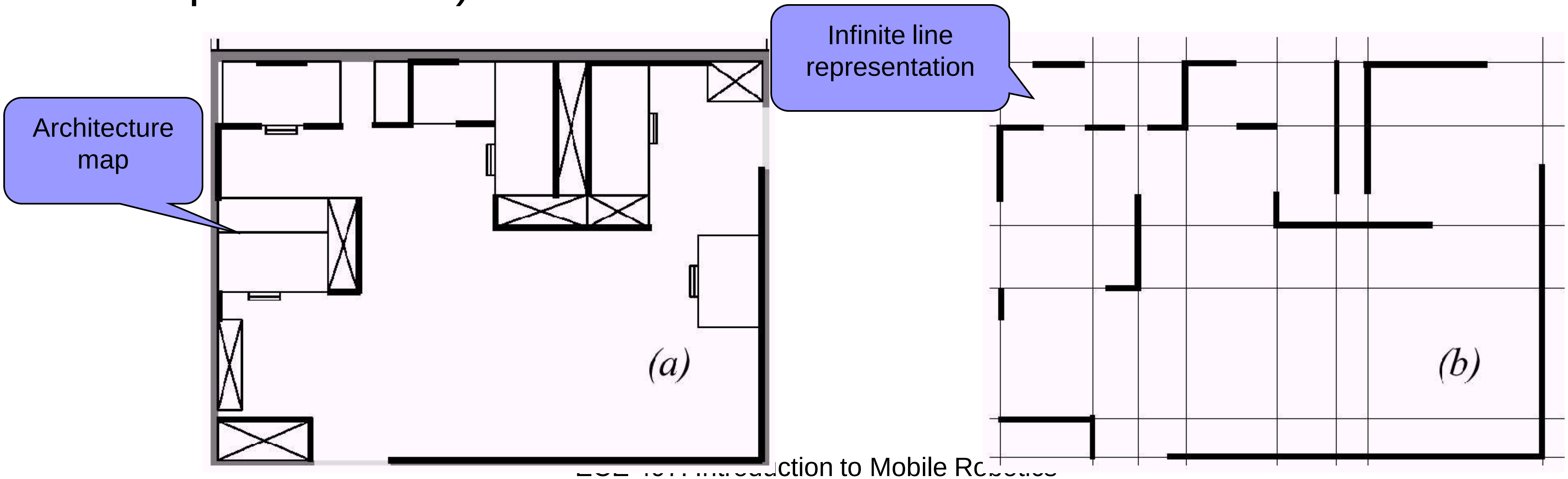
Continuous representation (5.5.1)

- a low-memory map is a 2D representation in which polygons represent all obstacles
- many simulations run exclusively in the computer memory and polygons are not used to describe a real-world environment
- When real environments must be captured, there are trends for *selectivity* and *abstraction*
  - □ *The human captures only objects that can be detected by the robot's sensors*
  - □ *This represents a subset of the features of the real world objects*



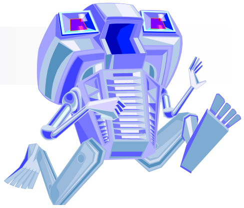
Continuous representation (5.5.1)

- *Geometric maps* represent the physical locations of objects without referring to their texture, color, elasticity, or any other secondary features that does not relate to position and space
- Memory usage can be reduce by capturing object geometry relevant to localization (i.e. continuous-valued line representation)



Architecture map

Infinite line representation

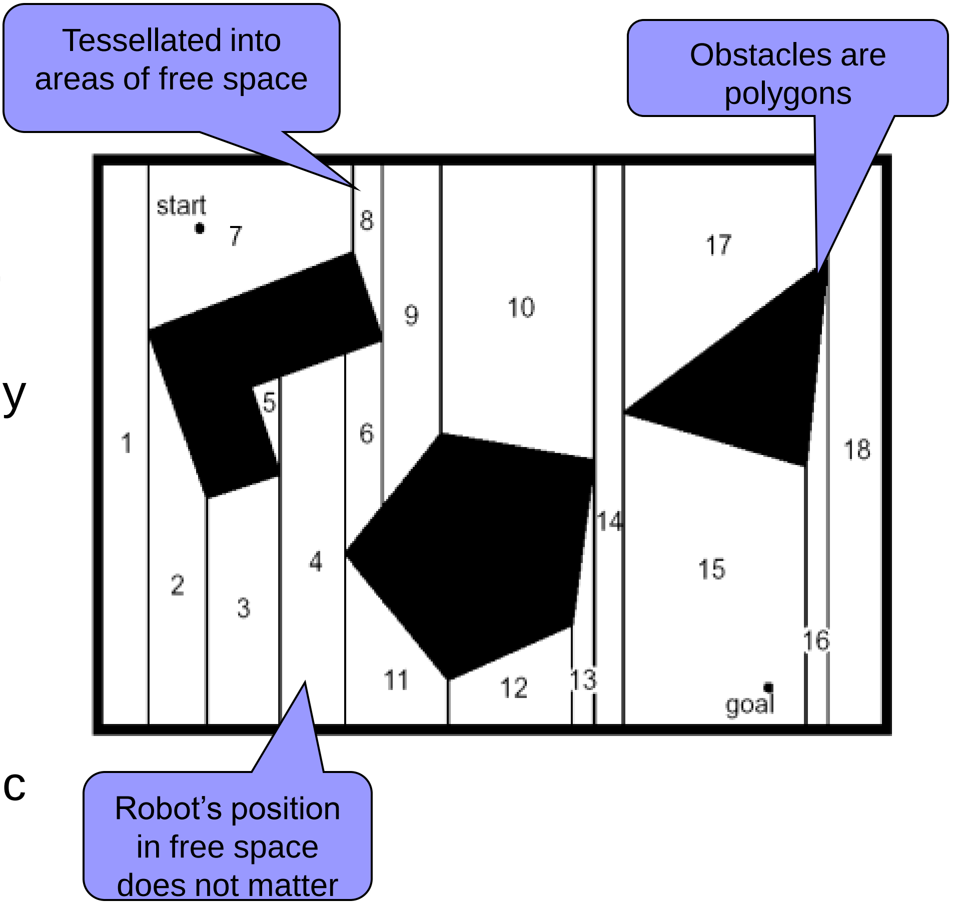


Decomposition strategies (5.5.2)

- One method of *simplification* is to approximate the real world environment lines as a set of infinite lines
- A more dramatic form of simplification is *abstraction*
  - *A general decomposition and selection of environmental features*
- The immediate disadvantage is the loss of fidelity between the map and the real world
- It may be useful if planned carefully to capture relevant, useful features of the world while discarding all other features

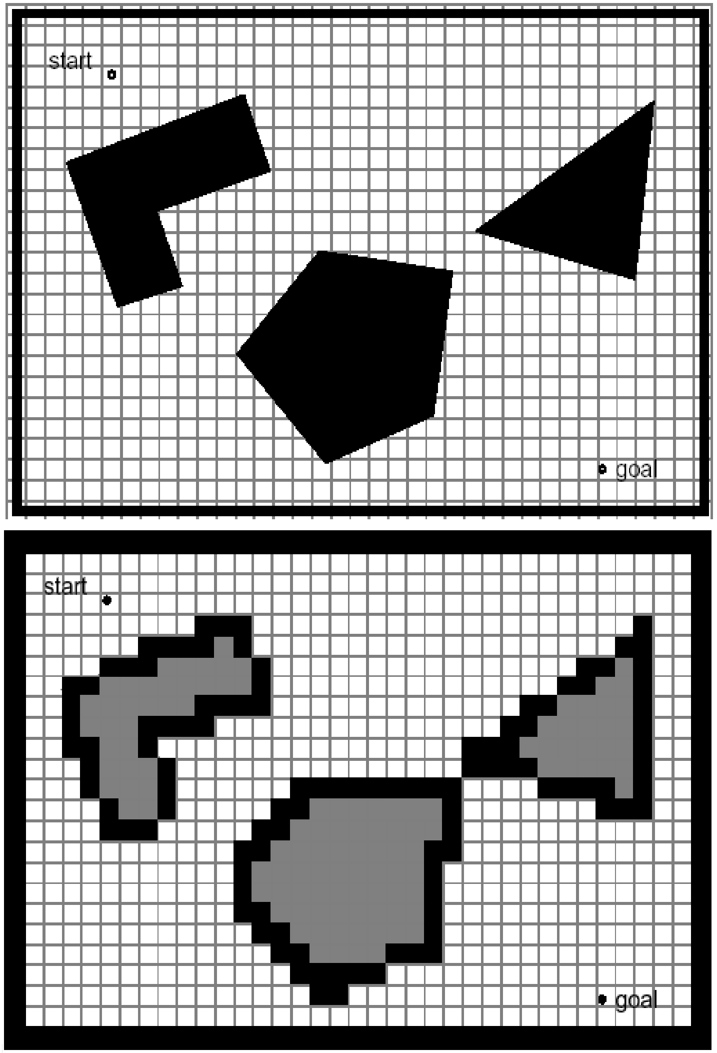
Decomposition strategies (5.5.2)

- Advantage:
  - the map representation is minimized
  - With hierarchical decomposition, reasoning and planning may be computationally superior to a fully detailed world model
  - A standard, lossless form of *opportunistic decomposition* is termed *exact cell decomposition* selects boundaries between discrete cells based on geometric criticality


Tessellated into areas of free space
Obstacles are polygons
start
goal
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
Robot's position in free space does not matter

Fixed Cell Decomposition (5.5.2)

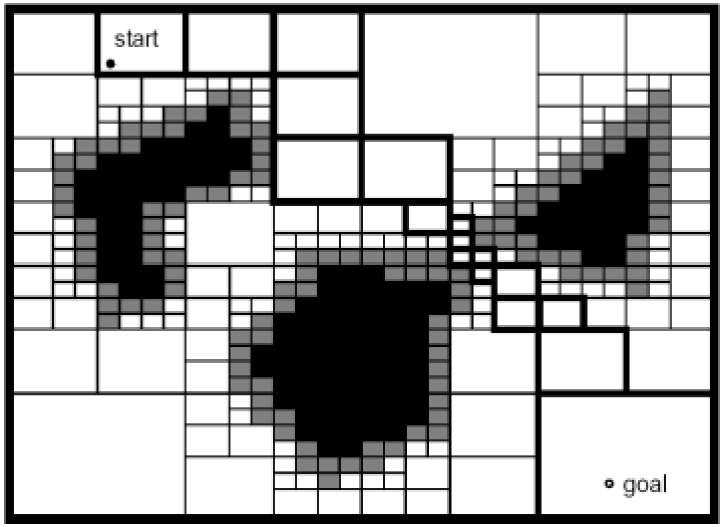
- *In fixed cell decomposition*, the world is tessellated into a discrete approximation of the continuous map
- The key disadvantage is the inexact nature
- Narrow passages are lost in this transformation



Adaptive Cell Decomposition (5.5.2)

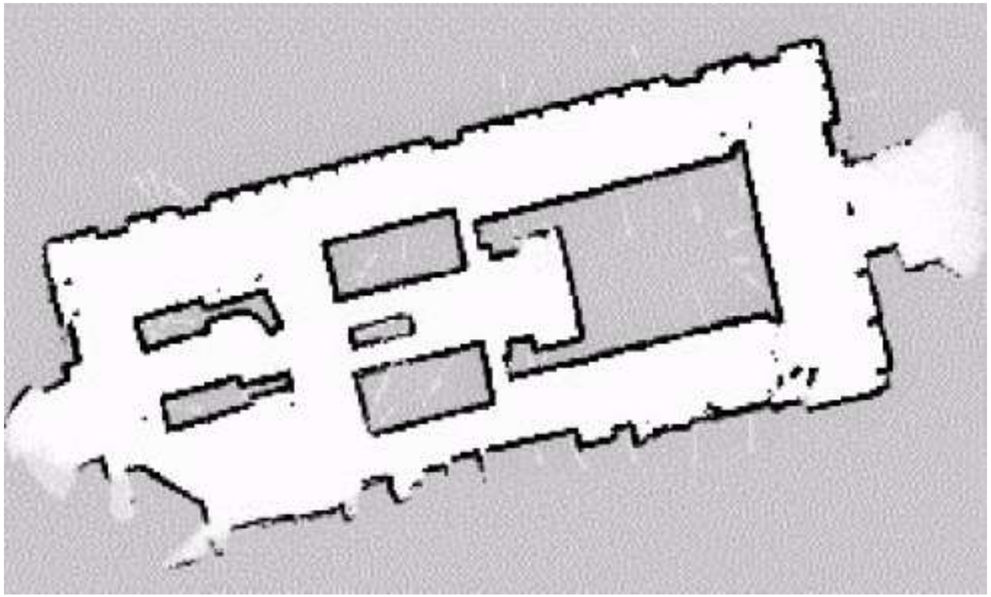
- Adaptive cell decomposition is extremely popular and the most common map representation in mobile robotics
- One version is called *occupancy grid* representation
- Every cell is either filled (part of an obstacle) or empty (part of free space)

White cells are outside the obstacles
Black cells are inside the obstacles
Gray cells are part of both regions



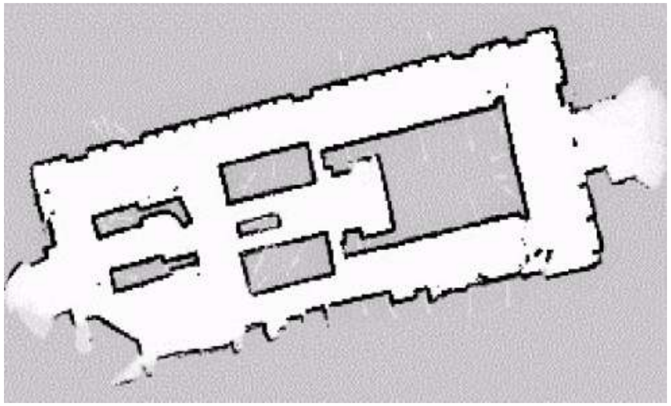
Occupancy Grid Map representation (5.5.2)

- A counter is used to determine how many times a cell is hit by a ranging sensor
- As the counter is incremented, the cell is deemed an obstacle
- The darkness of the cell is proportional to the value of the counter



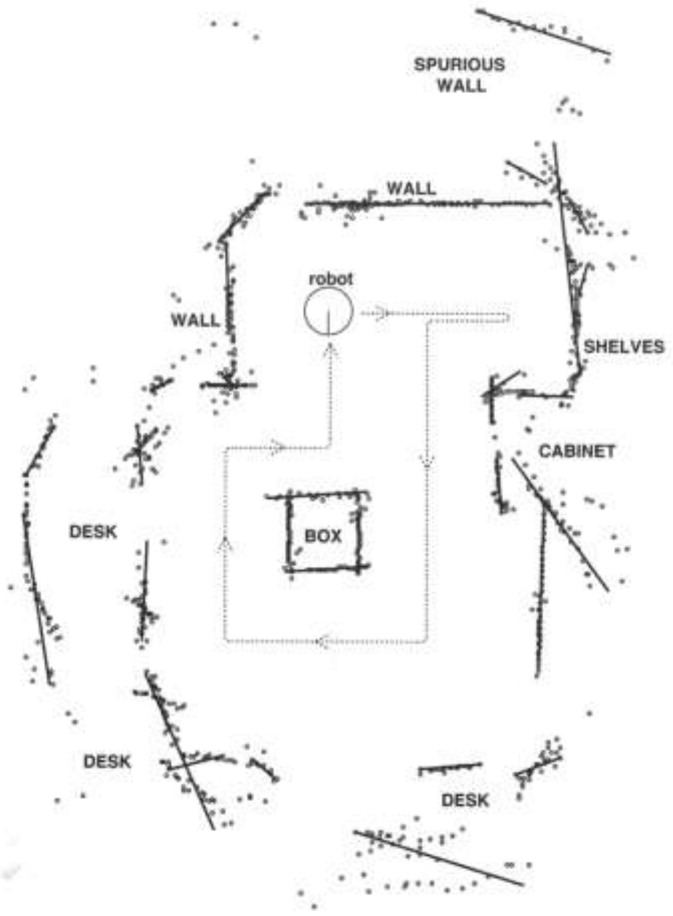
Occupancy Grid Map representation (5.5.2)

- Disadvantages:
  - The size of the map in robot memory grows with the environment size
  - Small cell sizes make the size of the memory untenable
  - Not compatible with the closed-world assumption which enables large, sparse environments to have small memory requirements
  - Imposes a geometric grid on the world a priori, regardless of environment details



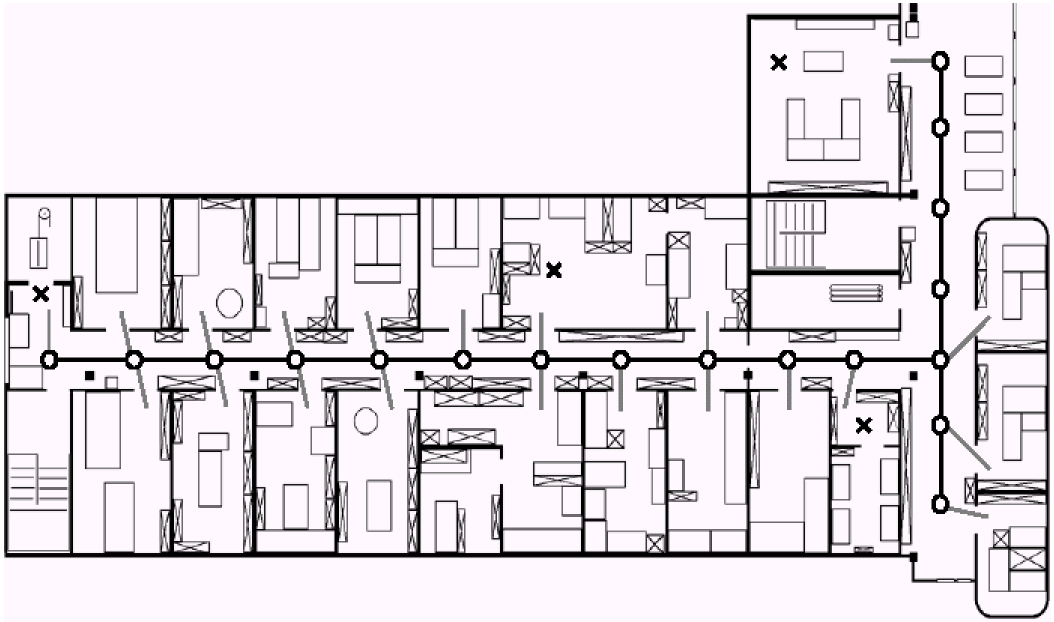
Occupancy Grid

- Created with sonar data
- Each cell is either occupied or unoccupied



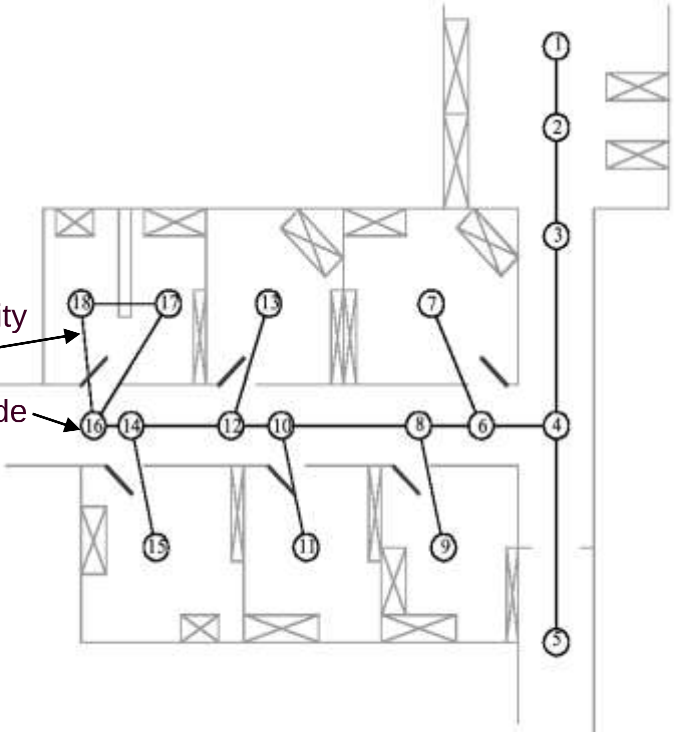
Topological Decomposition (5.5.2)

- Avoid direct measurement of geometric environmental qualities
- Concentrates on characteristics of the environment that are most relevant to the robot for localization



Topological Decomposition (5.5.2)

- Topological representations is a graph that specifies
  - *Nodes*
    - Areas in the world
  - *Connectivity arcs*
    - Denotes adjacent pairs of nodes
  - Adjacency is at the heart of the topological approach
  - Nodes are not of a fixed size or specifications of free space
  - Nodes document an area based on ay sensor discriminant

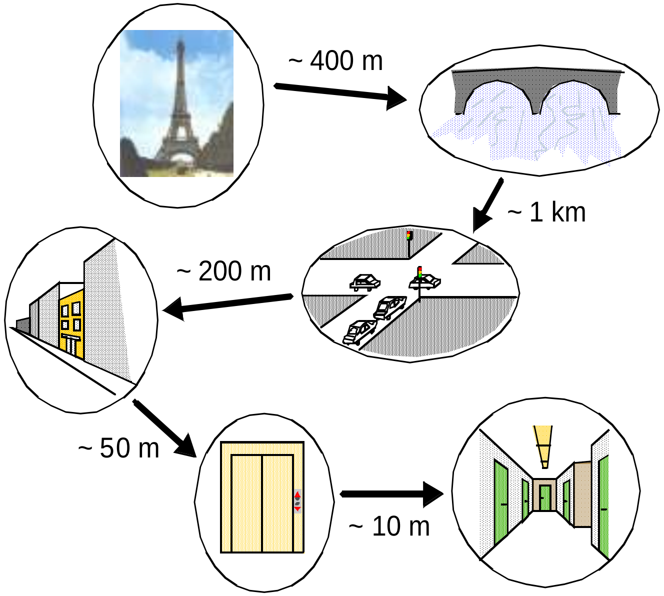


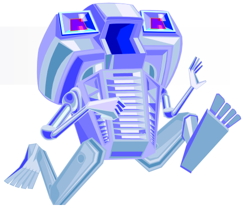
connectivity arcs

node

Topological Decomposition (5.5.2)

- To navigate a topological map robustly, a robot must satisfy 2 constraints
  - It must have means for detecting its current position in terms of the nodes of the topological graph
  - It must be able to travel between nodes using robot motion
- Node sizes and dimensions must be optimized to match the sensory discrimination of the mobile robot hardware



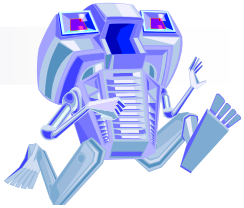


Topological Map

- Store what the robot needs to do at each landmark
  - *Landmark-based map*
- The map can be stored (represented) in different forms
  - Store all possible paths and use the shortest one
  - *Topological map*
    - describes the connections among the landmarks
  - *Metric map*
    - global map of the maze with exact lengths of corridors and distances between walls, free and blocked paths
- The robot can use this map to find new paths through the maze
- Such a map is a *world model*, a representation of the environment

World Models

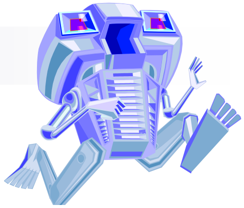
- Numerous aspects of the world can be represented
  - □ self/ego
    - stored proprioception, self-limits, goals, intentions, plans
  - □ *space*
    - metric or topological (maps, navigable spaces, structures)
  - □ *objects, people, other robots*
    - detectable things in the world
  - □ *actions*
    - outcomes of specific actions in the environment
  - □ *tasks*
    - what needs to be done, in what order, by when
- Ways of representation
    - Abstractions of a robot’s state & other information



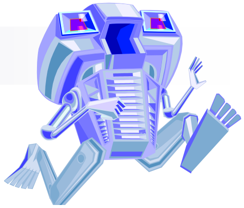
Model Complexity

- Some models are very elaborate
  - □ They take a long time to construct
  - □ These are kept around for a long time throughout the lifetime of the robot
  - □ E.g.: a detailed metric map
- Other models are simple
  - □ Can be quickly constructed
  - □ In general they are transient and can be discarded after use
  - □ E.g.: information related to the immediate goals of the robot (avoiding an obstacle, opening of a door, etc.)

Models and Computation

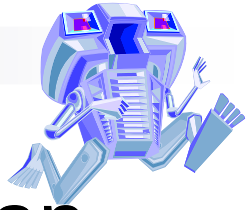


- Using models require significant amount of computation
- *Construction*
  - the more complex the model, the more computation is needed to construct the model
- *Maintenance*
  - models need to be updated and kept up-to-date, or they become useless
- *Use of representations*
  - complexity directly affects the type and amount of computation required for using the model
- Different architectures have different ways of handling representations



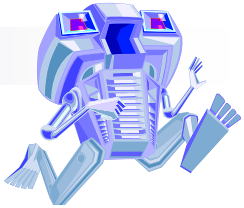
Metric Maps

- *Construction*
  - □ Requires exploring and measuring the environment and intense computation
- *Maintenance*
  - □ Continuously update the map if doors are open or closed
- *Utilization*
  - □ Finding a path to a goal involves planning: find free/navigational spaces, search through those to find the shortest, or easiest path



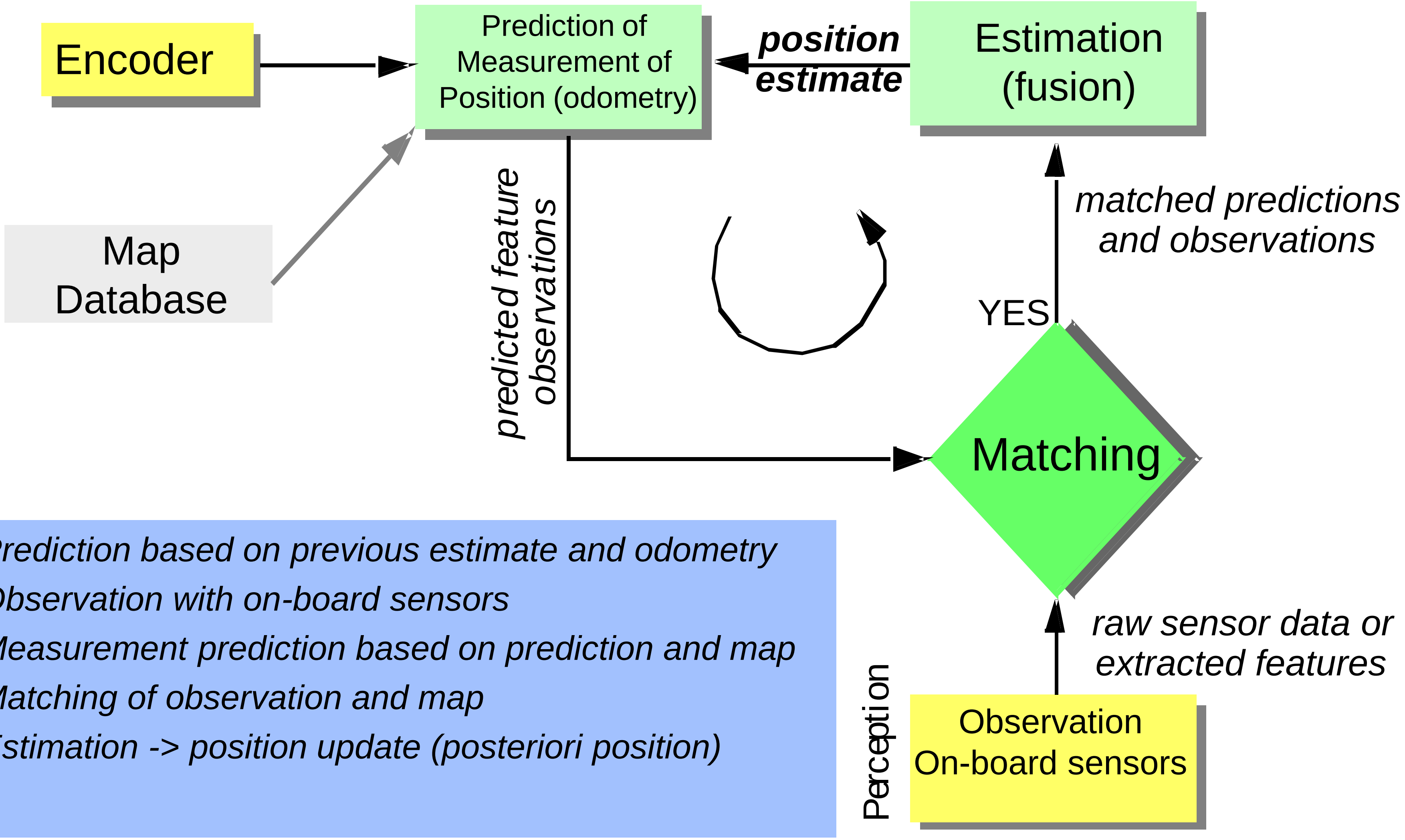
State-of-the-Art: Current Challenges in Map Representation (5.5.3)

- The real world is dynamic
- Cannot distinguish between permanent and transient obstacles
- 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



Probabilistic Map-Based Localization (5.6)

The Five Steps for Map-Based Localization


Encoder
Prediction of Measurement of Position (odometry)
position estimate
Estimation (fusion)
Map Database
predicted feature observations
matched predictions and observations
YES
Matching
raw sensor data or extracted features
Perception
Observation On-board sensors

1. Prediction based on previous estimate and odometry

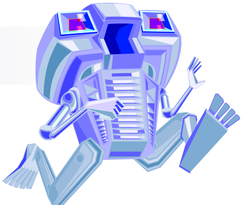
2. Observation with on-board sensors

3. Measurement prediction based on prediction and map

4. Matching of observation and map

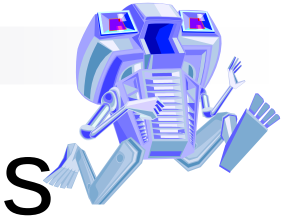
5. Estimation -> position update (posteriori position)

Probabilistic Map-Based Localization (5.6.1)

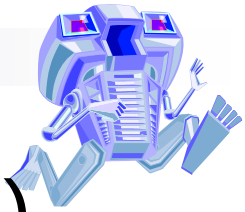


- One geometric approach to *multi-hypothesis representation* identifies the possible positions of a robot
- *Probabilistic techniques* identifies probabilities with the possible robot positions
- Two classes of probabilistic localization are:
  - *Markov localization*
  - *Kalman filter localization*

Probabilistic Localization Classes (5.6.1)

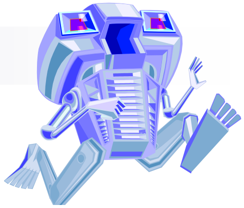


- *Markov localization*
  - □ Uses an explicitly defined probability distribution across all robot positions
- *Kalman filter localization*
  - □ Uses a Gaussian probability density representation of robot position and scan matching for localization
  - □ Unlike Markov, it does not independently consider each possible robot pose
  - □ Kalman results from the Markov axioms if the robot's position uncertainty is assumed to be Gaussian



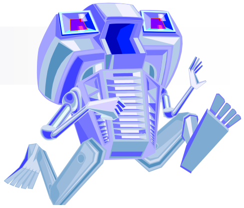
Probabilistic Map-Based Localization (5.6)

1. Consider a mobile robot moving in a known environment.
2. As it starts to move from a precisely known location, it might keep track of its location using odometry.
3. Due to odometry uncertainty, after some time the robot will get very uncertain about its position.
4. To keep this uncertainty from growing unbounded, the robot must localize itself in relation to its environment map
5. The robot uses onboard sensors to make observations about its environment
6. Information from odometry and the exteroceptive observations can be combined for the robot to localize itself



Probabilistic Map-based Localization (5.6.1)

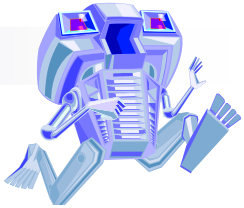
- The process of updating robot position based upon *proprioceptive* and *exteroceptive* sensor values are separated logically into a general two-step process
  - *Action Update*
    - Proprioceptive
    - Represents the application of some action model
  - *Perception Update*
    - Exteroceptive
    - Represents the application of some perception model



Action Update (5.6.1)

- Application of some action model, *Act* to the mobile robot's *proprioceptive* encoder measurements o_t and prior belief state s_{t-1} to yield a new belief state, s_t , representing the robot's belief about it's current position

$$\mathbf{s_t = Act\ (o_t, s_{t-1})}$$

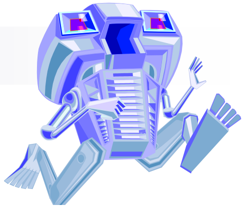


Perception Update (5.6.1)

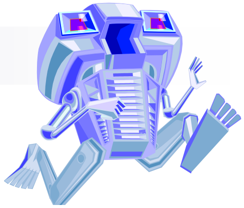
- Application of some perception, *See* to the mobile robot's *exteroceptive* sensor inputs i_t and updated belief state s_t to yield a refined belief state, s_t, representing the robot's belief about it's current position

$$\mathbf{s_t = See\ (i_t, s_t\ )}$$

Action versus Perception Update (5.6.1)

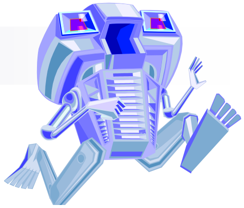


- The perception model See and sometimes the action model Act are abstract functions of both map and the robot’s physical configuration
- The *action update* contributes uncertainty to the robot’s beliefs about position because encoders have errors
- The *perception update* generally refines the belief state because sensors provide clues about the robot’s possible position



Markov Localization (5.6.1)

- *Markov localization* is the robot’s belief state usually represented as separate probability assignment for every possible pose on the map
  - □ Special case of probabilistic state estimation applied to mobile robot localization
- *Kalman filter localization* represents the robot’s belief state using a single, well-defined Gaussian probability density function
  - □ It retains a μ and σ parameterization of the robot’s belief about position with respect to the map

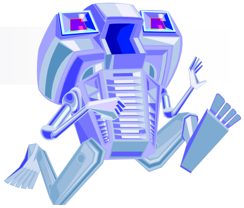


Markov vs. Kalman (5.6.1)

- Markov
  - Allows localization starting from any unknown position
  - Recovers from ambiguous situations because the robot can track multiple, complete disparate possible positions
  - Requires discrete representation of the space (geometric grid or topological graph)
  - Required memory and computational power can limit precision and map size

- *Kalman*
  - Tracks the robot from a known position
  - Is both precise and efficient
  - Can be used in continuous world representations
  - If robot uncertainty becomes too large and not unimodal, it can fail to capture the multitude of possible robot positions and can become irrevocably lost

Markov Localization (5.6.2)



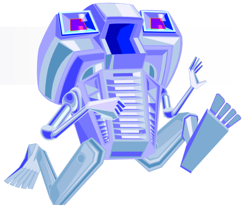
Markov Localization (5.6.2)

- Implements the generic belief representation by tessellating the robot configuration space into a finite, discrete number of robot poses in the map
- During each update, the belief state is computed that results when new information (encoder and sensor values) are incorporated into a prior belief state with an arbitrary probability density
- the probability theory of the solution is based upon *Bayes formula*

Probability theory to robot localization (5.6.2.1)

- Given a discrete representation of robot positions, assign a probability that the robot is indeed at that position, p(A)
  - □ p(A) – prior probability of A
  - □ Measures the probability that A is true independent of any additional knowledge we may have
  - □ $p(r_t = \ell)$ – prior probability that robot r is at position ℓ at time t
- To compute the probability given the encoder and sensor evidence
  - □ p(A|B) – conditional probability of A given that we know B
  - □ $p(r_t = \ell | i_t)$ – prior probability that robot r is at position ℓ at time t given hat the robot's sensor inputs i

Markov Localization (5.6.2):
Bayes Rule



- The product rules states that the probability that A and B are both true is given by :

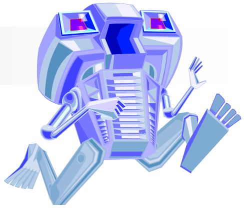
$$p(A \wedge B) = p(A|B)p(B)$$
$$p(A \wedge B) = p(B|A)p(A)$$

- From these expressions, *Bayes rule* is:

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

- *Bayes rule* is used to determine the robot's new belief state as a function of its sensory inputs and its former belief state

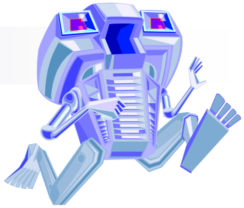
Markov Localization (5.6.2): Bayes Rule



- The *See* function expresses a mapping from a belief state and sensor input to a refined belief state. Update the probability associated with each position ℓ in L. *p(ℓ) = p(r = ℓ)*

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

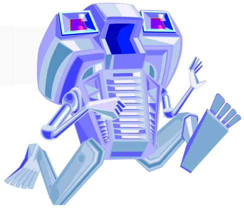
- *p(i|ℓ)* is the probability of a sensor input at each robot position and it must be computed from some model
- *p(ℓ)* is the probability that the robot’s belief state is at l before the perceptual update process
- *p(i)* does not depend on ℓ and is a constant and is usually dropped and at the end of the perception update, all probabilities in the belief state are re-normalized to sum to 1.0



Markov Localization (5.6.2): Bayes Rule

- The *Act* function maps a former belief state and encoder measurement (i.e. robot action) to a new belief state.
- To compute the probability of position l in the new belief state, integrate or sum all possible ways in which the robot may have reached l.
- The same location can be reached from multiple source locations with the same encoder measurement, *o*.

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



Markov Localization (5.6.2): Markov assumption

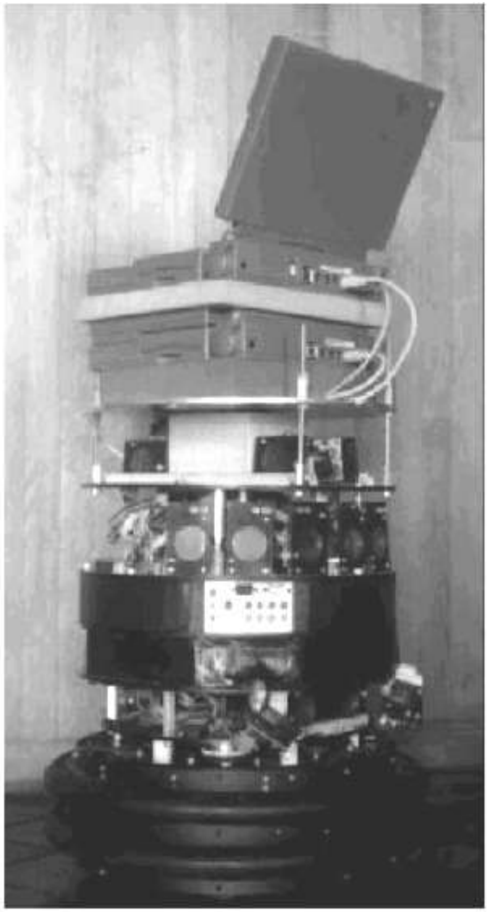
- The Act and See equations from the basis of Markov localization and incorporate the *Markov assumption*
  - □ The outputs are a function only of the robot's previous state and its most recent actions (odometry) and perception
  - □ The assumption may not always be valid but it greatly simplifies tracking, reasoning and planning and it is an *approximation*

Markov Localization (5.6.2.2): Case Study 1 - Topological Map

- Markov localization is possible when the environment provides an appropriate decomposition (i.e. topological)
- Each robot receives a topological description of the environment (i.e. connectivity of hallway and rooms, no geometric information) [AAAI 1994]
- Map contains several false arcs
- Robot was to move the map to navigate from a starting position to a target room
- The Dervish Robot used probabilistic Markov localization and a multiple-hypothesis belief state

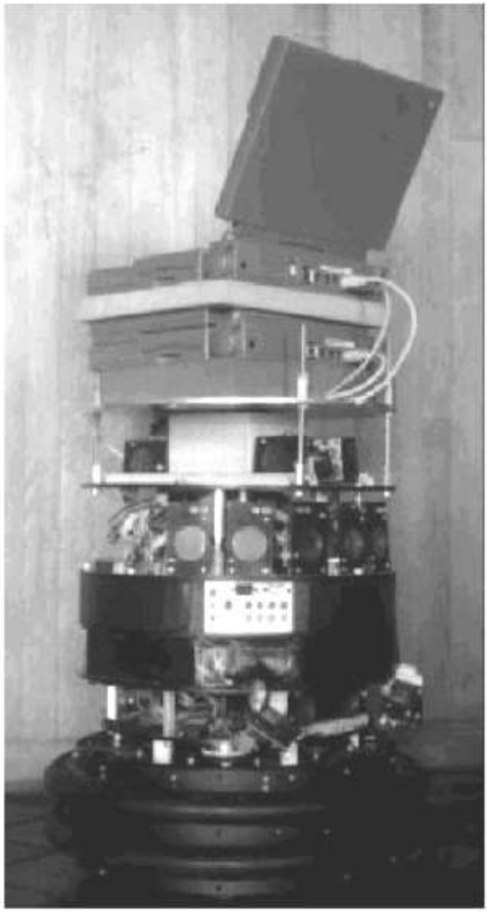
Markov Localization (5.6.2.2): Case Study 1 – Robot Design

- Traditional sonar were arranged radially around the robot in a ring
  - Disadvantage is that it makes robot subject to tripping over short objects and being decapitated for tall objects
- One pair of sonar were diagonally upward to detect ledges
- One pair of sonar were mounted on the base to detect low obstacles
- Sonar were grouped to reduce crosstalk
- Dervish’s perceptual system was used to detect matching perceptual events (the detection and passage of connections between hallways and offices)



Markov Localization (5.6.2.2): Case Study 1 – Perceptual System

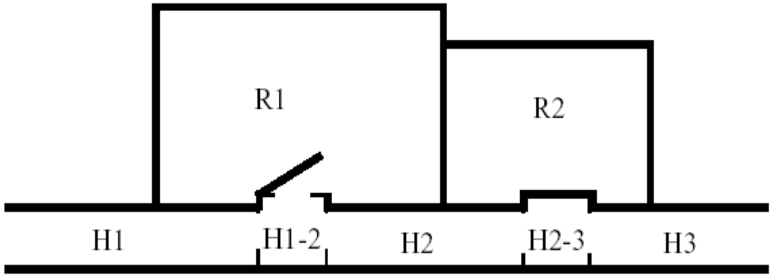
- The perceptual system was *abstract* and used the trajectory of sonar strikes to the left and right of the robot over time
  - There was no use of encoder values to trigger perceptual events
  - If the robot detected a 7 to 17 cm indentation in width for more than a second continuously then a *closed door sensory event* was triggered
  - If the sonar strikes were beyond 17 cm for more than a second then an *open door sensory event* was triggered
- When the angle to the robot center line exceeded 9 degrees, the sensory events were suppressed
  - These false negatives suggested a probabilistic solution to the localization problem in order to compute a complete trajectory of perceptual inputs



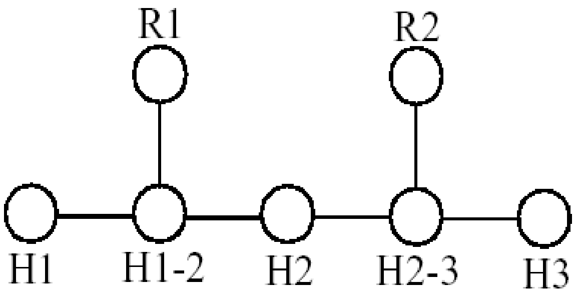
Markov Localization (5.6.2.2): Case Study 1 - Topological Map

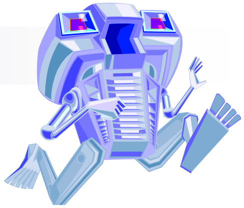
- Dervish used a *discrete topological map*
  - □ Identical in abstraction and information to the contest map
  - □ Decision involves assignment of nodes and connectivity between nodes
  - □ Node boundaries are marked by doorways, hallways, and foyers
  - □ Note there is no geometric information on the nodes

Contest environment map



Topologic map

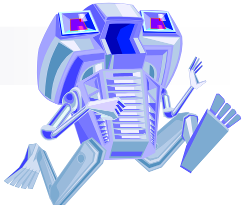




Markov Localization (5.6.2.2): Case Study 1 – Belief State

- In order to represent a specific belief state,
  - □ For each topological node, n, there was a *probability or likelihood* that the robot is at a physical position within the boundaries of n. $p(r_i = n)$
  - □ The probabilities were approximate thus they were *likelihoods*
- The perception update were generated asynchronously each time the feature extractor recognized a large scale feature (e.g., doorway, intersection)
  - □ Each perceptual event consists of a percept-pair (a feature on one or both sides of the robot)
- From equation 5.21, p(n) represents the current belief state of Dervish. The challenge lies in computing $p(i|n)$

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



Markov Localization (5.6.2.2): Case Study 1 – Certainty Matrix

- Because the feature extraction only extracts 4 total features (nothing, closed door, open door, open hallway) and a node contains one of 5 total features (wall, closed door, open door, open hallway, foyer)
- These 4 x 5 possible combinations can be represented in a *lookup table*
- This lookup table is a *certainty matrix*
- The probability is a function of the feature extracted and the actual feature in the node
- The human generates a specific certainty matrix that represents the robot's perceptual confidence along with a global measure for the probability that any given door is closed versus open in the real world
- The probability that the robot is next to an *open hallway* and recognizes it as an *open door* is *0.10*

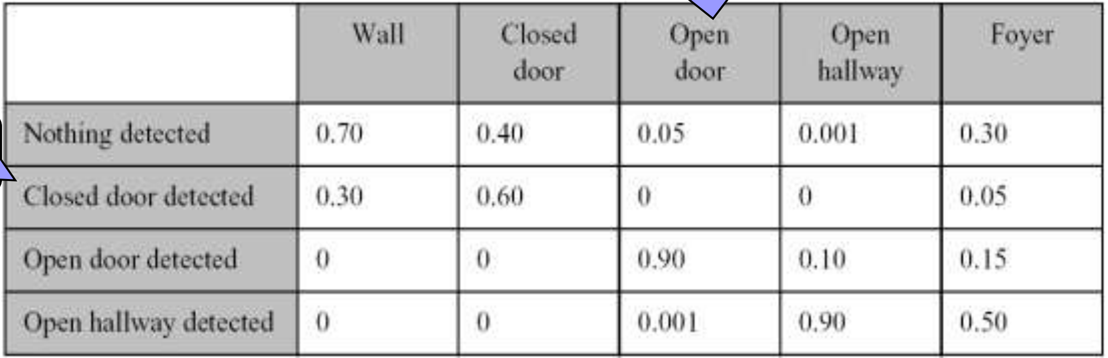
n =open hallway

i = open door

p(i|n) = 0.10

Node feature or world feature

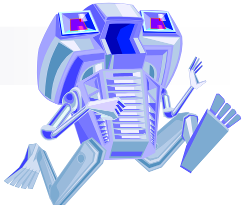
Extracted feature



	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

Markov Localization (5.6.2.2): Case Study 1 – Perception Update (1)

- Dervish has no encoders and perceptual events are triggered asynchronously by the feature extraction process *(no action update)*
  - However, the robot is moving and therefore we can apply a combination of action and perception update
  - It may take several perceptual events to update the likelihood of every possible robot position given Dervish’s former belief state
- The perception update formula is a combination of the general form of action update and perception update



Markov Localization (5.6.2.2):
Case Study 1 – Perception Update Formula

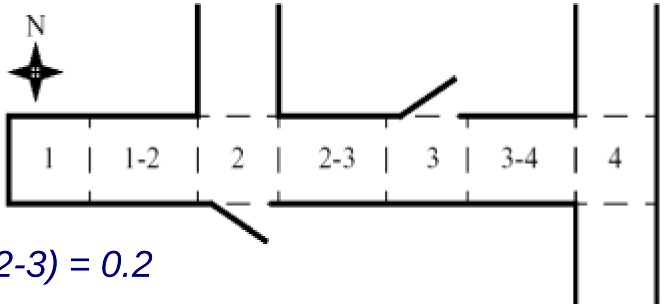
- The likelihood of position *n* given perceptual event *i* or the update of belief state for position *n* given the percept-pair *i* is calculated by the following

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

  - p(n $_{t-i}$) is the likelihood of being at position *n* given the former belief state
  - t-i is used instead of t-1 because the topological distance between n’ and n can vary depending on the specific topological map
  - *p(n_t|n $_{t-1}$, i_t)* is calculated by multiplying the probability of generating a perceptual event *i* at position *n* by the probability of having failed to generate events at all nodes between *n* and *n*

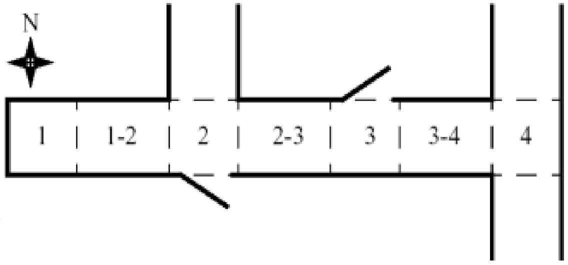
$$p(n_t|n'_{t-i}, i_t) = p(i_t, n_t)\cdot p(\varnothing, n_{t-1})\cdot p(\varnothing, n_{t-2})\cdot \ldots \cdot p(\varnothing, n_{t-i+1})$$

Markov Localization (5.6.2.2): Case Study 1 – Example Calculation



- For the following topological map, make 2 assumptions:
  - the robot is facing east
  - the robot has two nonzero belief states, *p(1-2) = 1.0 and p(2-3) = 0.2*
  - The probability that any given door is closed versus open is *p(closed door) = 0.60*
- Suppose that the robot detects an *open hallway* on the left and an *open door on the right* simultaneously
- State 2-3 will progress potentially to 3, 3-4, or 4
- States 3 and 3-4 can be eliminated because the likelihood of detecting an open door when there is only a wall is zero *p(door|wall) = 0.0*.
- The likelihood of reaching state 4 is the product of
  - the initial likelihood *p(2-3)= 0.2*
  - the likelihood of not detecting anything at node 3 *(a)*
  - the likelihood of detecting a hallway on the left and a door on the right at node 4 *(b)*
  - (for simplicity we assume that the likelihood of detecting nothing at node 3-4 is 1.0)

Markov Localization (5.6.2.2): Case Study 1 – Example Calculation (2)



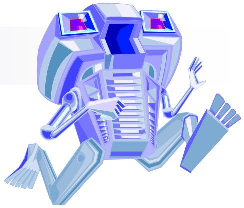
- If Dervish detects nothing at node 3 then
  - he failed to detect the door (open or closed) on its left
    - *p(nothing|closed door)· p(closed door) = (0.40)(0.60)*
    - *p(nothing|open door)·p(open door) = (0.05)(1 – 0.60)*
  - and correctly detects nothing on its right, *p(nothing|wall) = 0.7*
- If Dervish detects at node 4
  - the hallway on the left *p(hallway|hallway) = 0.90* and
  - mistakenly identifies an open door on the right *p(open door|hallway) = 0.10*
- The final formula becomes

p(4) = p(2-3)·p(nothing|door) ·p(nothing|wall) ·p(hallway|hallway) ·p(open door|hallway)

p(4) = 0.2·[(0.6)(0.4) + 0.4)(0.05)] ·0.7 · [0.9 · 0.1] = .003276

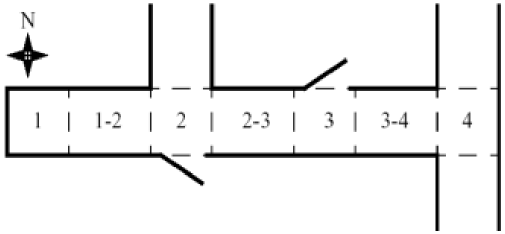
which represents a *partial belief state for node 4* given the prior *belief state 2-3*

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

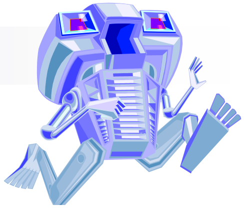


Markov Localization (5.6.2.2):
Case Study 1 – Example Calculation (3)

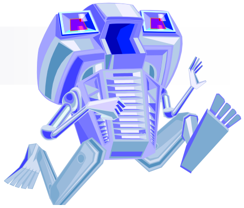
- What if the robot’s prior belief state is at *node 1-2*
- The robot will potentially progress to 2, 2-3, 3, and 3-4
- States 2-3, 3 and 3-4 can be eliminated because the likelihood of detecting an open door when a wall is present is zero *p(door|wall) = 0.0*.
- p(2) = p(1 – 2) · p(open door|right door) · p(hallway|left hallway) = 1.0 · [0·0.6 +0.90· 0.4] · 0.90 = *0.324*
- applying the progression to node 4 from 1-2 yields *p(4) = p(2) · 4.3 · 10*$^{-6}$ *= 1.3932· 10*$^{-6}$ which represents a the *belief state for node 4* given the prior *belief state 1-2*
- *the total belief state for node 4 = p(4:2-3) + p(4|1-2) = 0.003276 + 1.3932· 10*$^{-6}$ = 0.003277



Markov Localization (5.6.2.2):
Case Study 1 – Topological map

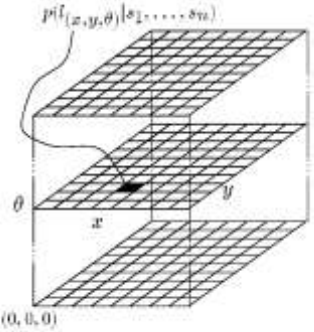


- Dervish was successfully able to navigate four different indoor office environments with no notion of the distance between adjacent nodes in its topological map
- This demonstrates the power of probabilistic localization in spite of a lack of action and encoder information
- Question:
  - how does the robot decide how to move, given that it has multiple possible robot positions in its representation?
  - plan the robot's actions by assuming that the robot's actual position is its most likely node in the belief state
  - generally the most likely position is a good measure of the robot's actual world position
- One step to improve the planning system is to specify a *goal belief state* than a goal position
  - the robot can reason and plan in order to achieve a goal confidence level
  - the robot takes into account not only the position but the measured likelihood of each position

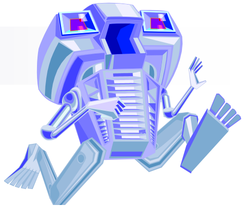


Markov Localization (5.6.2.3): Case Study 2 – Grid Map

- The major weakness of a topological decomposition is the *resolution limitation*
- A more precise navigation uses a grid-based representation while still employing the *Markov localization technique*
- This case study used Rhino, a RWI B24 robot with 24 sonar and 2 Sick laser rangefinders
- Rhino uses a 2D geometric environmental representation of free and occupied space
- This map is tessellated regularly into a *fixed decomposition* grid
- Rhino uses a multiple-hypothesis belief state
  - Rhino consists of a 15 x15 x 15 3D array representing 15^3 possible robot positions
  - The resolution of the array is 15 cm x 15 cm x 1°
  - Unlike Dervish which assumes the orientation is approximate and known, Rhino explicitly represents alternative orientations
  - Rhino's belief state has 3 degrees of freedom
  - Rhino includes encoder inputs, metric distance and both and explicit actior update phase and perception update phase



Markov Localization (5.6.2.3): Case Study 2 – Action and Perception Update



- Action update:
  - □ Due to the tessellated representation of position, the discrete Markov chain of the action update was performed
  - □ Given encoder measurements *o* at time *t*, each updated position probability in the belief state Is expressed as a sum over previous possible positions and motion model

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

- Perception update:
  - □ Given a range perception *i*, the probability of the robot being at each location *l* is

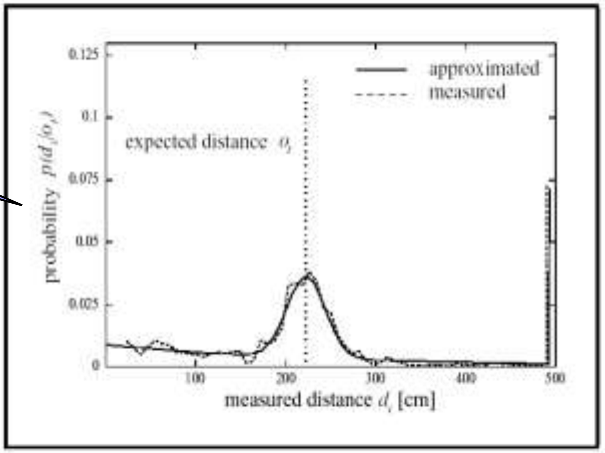
$$p(l|i) = \frac{p(i|l)p(l)}{p(i)}$$

- Unlike Dervish, the number of possible values for *i* and *l* cannot be recorded on a lookup table
- Rhino computes *p(i|l)* using a model of the robot's sensor behavior

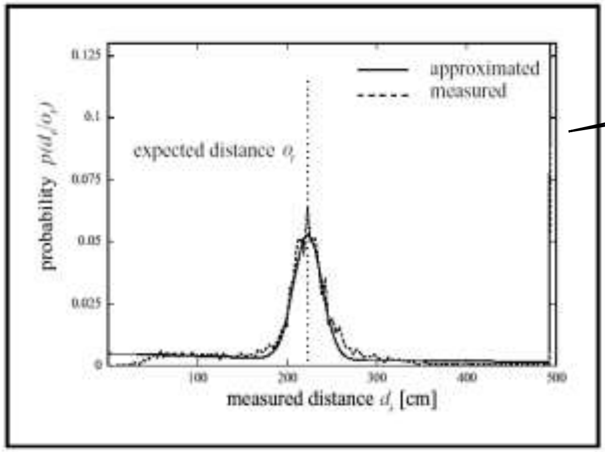
Markov Localization (5.6.2.3): Case Study 2 – Sensor Model

- The *sensor model* must calculate the probability of a specific perceptual measurement given that its likelihood is justified by known errors of the sonar or laser rangefinder
- Assumptions
  - □ Measurement error can be described by a distribution with a mean at the correct reading
  - □ Non-zero chance that a range sensor will read any measurement value
  - □ there will be a local peak in the probability density distribution at the maximal reading of a range sensor due to absorption or reflection failure mode

ultrasound

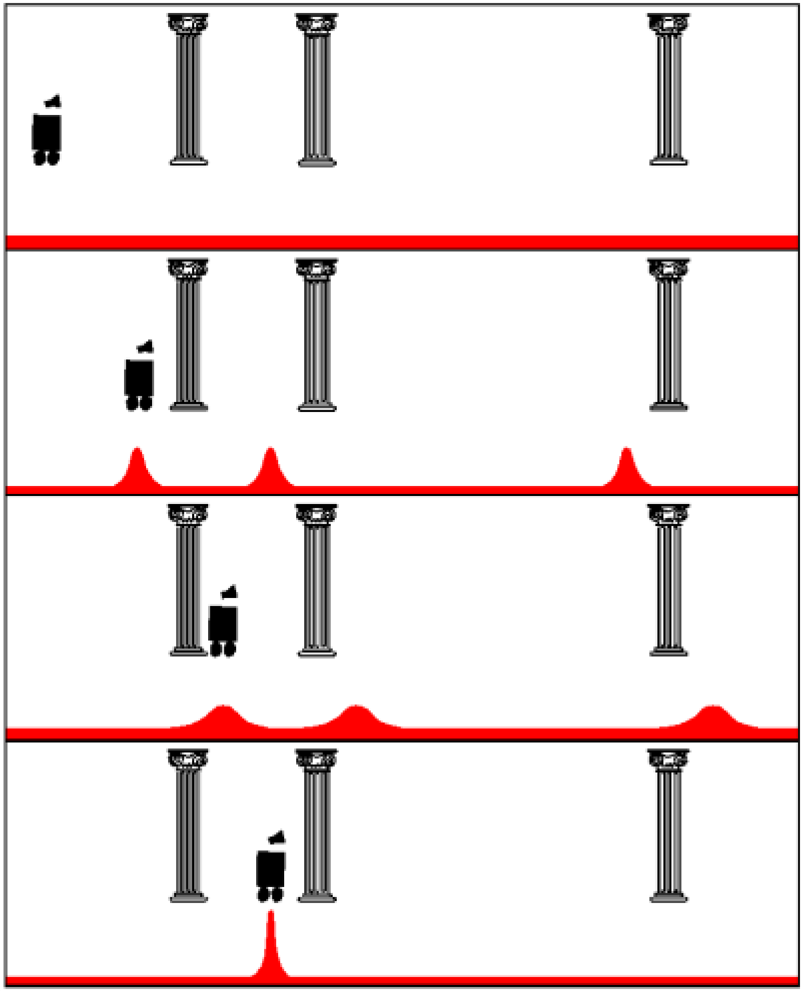


laser



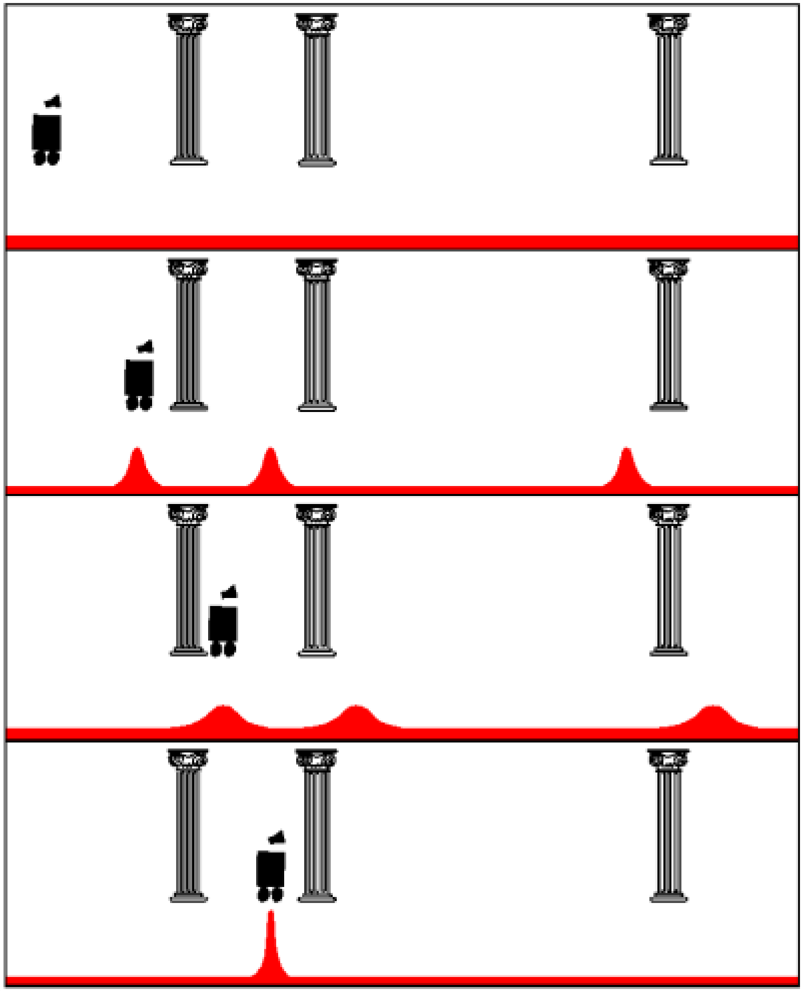
Markov Localization (5.6.2.3): Case Study 2 – Grid Map (1D example)

1. Start
   - No knowledge at start, thus there is a 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 the estimate of the new probability distribution based on the previous one and the motion.
4. Robot perceives second pillar
   - Based on all prior knowledge the probability being at pillar 2 becomes dominant

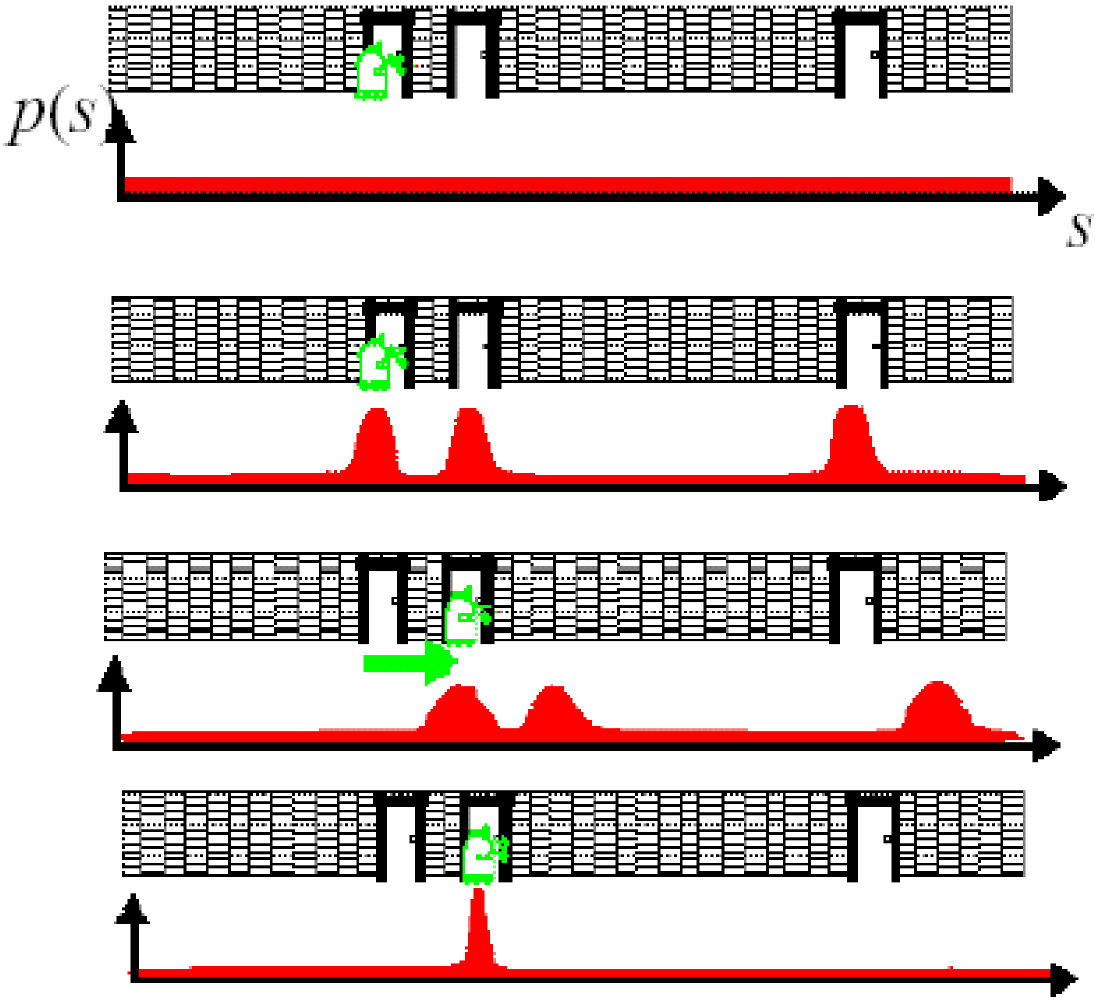


Markov Localization (5.6.2.3):
Case Study 2 – Grid Map (1D example)

- As the robot encounters one pillar and then a second pillar, the probability density function over possible positions becomes multimodal, unimodal and then sharply defined
- The ability of a Markov localization system to *localize the robot from an initially lost belief state* is its key distinguishing feature
- This is a challenging application because of the dynamic nature of the environment



Markov Localization Example



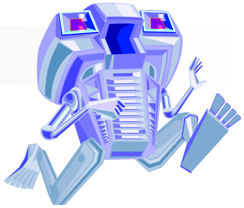
The robot is placed somewhere in the environment but it is not told its location

The robot queries its sensors and finds out it is next to a door

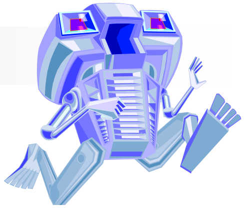
The robot moves one meter forward. To account for inherent noise in robot motion the new belief is smoother

The robot queries its sensors and again it finds itself next to a door

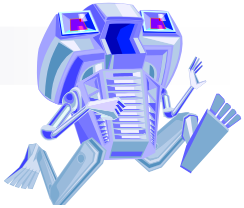
Markov Localization (5.6.2.3): Case Study 2 – Grid Map



- 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



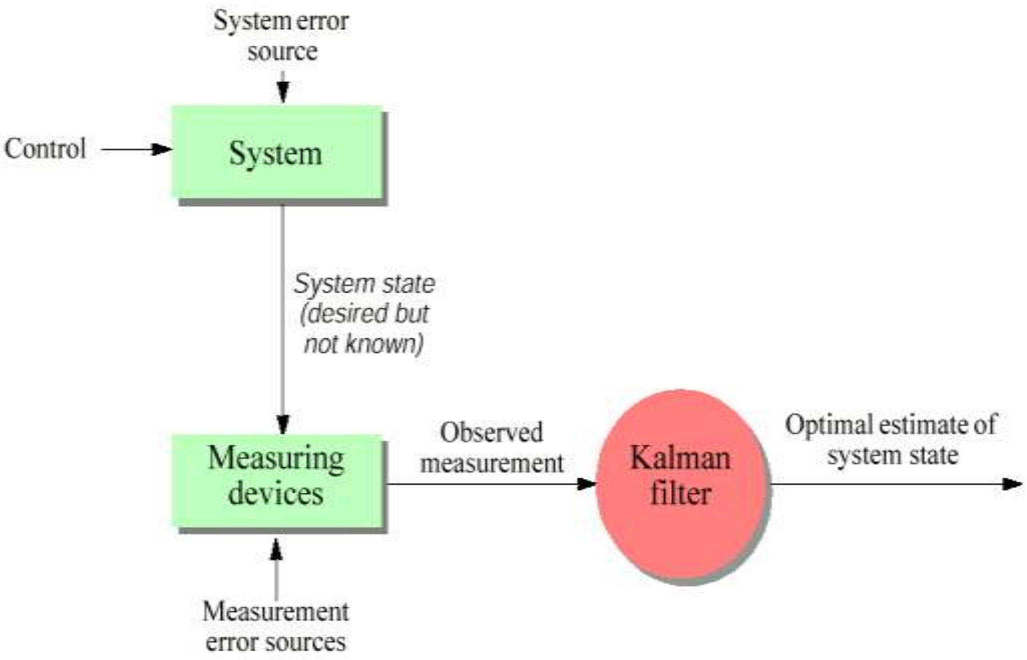
Kalman Filter Localization (5.6.3)



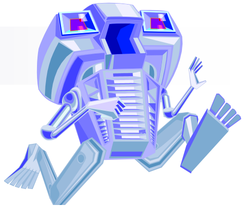
Kalman Filter Localization (5.6.3)

- The Markov localization model can represent any probability density function over robot position
- One can argue that not the probability density curve but the *sensor fusion* problem is key to robust localization
- Optimal localization should take into account the information provided by all of the heterogeneous sensors
- The *Kalman filter* is used to achieve sensor fusion
- The *Kalman filter* is more efficient than Markov localization
- The benefit of the simplification of the probability density function is a resulting *optimal recursive data-processing algorithm*
  - □ it incorporate all information, regardless of precision to estimate the current robot's position

Kalman Filter Localization (5.6.3): General Scheme


System error source
Control
System
System state (desired but not known)
Measuring devices
Observed measurement
Kalman filter
Optimal estimate of system state
Measurement error sources

- Inputs to the system are a control signal and system error sources
- *The Kalman filter* produces an optimal estimate of the system state based on the knowledge of the *system* and *the measuring device*
- *The Kalman filter* fuses sensor signals and system knowledge in an optimal way



Kalman Filter Localization (5.6.3): Kalman Filter Theory

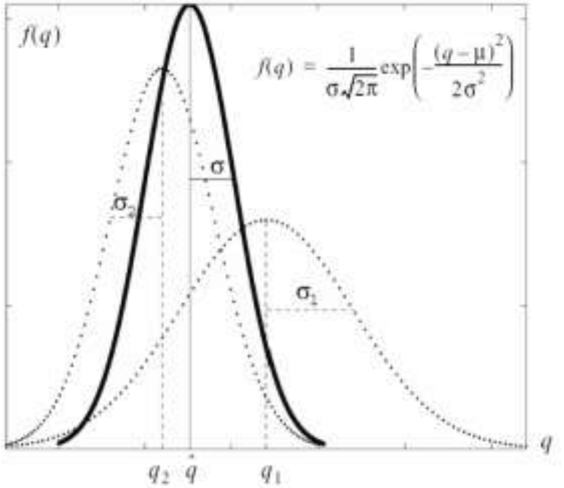
- Multiple measurements are incorporated into a single estimate of state
- Assume that the state does not change between the measurements
- This is referred to as *static estimation*
- Suppose the robot has ultrasonic and laser sensors
- the laser provides richer and more accurate data but suffers from failure such as detecting glass while the sonar will provide an accurate reading
- the sensor fusion is extremely efficient as long as the error characteristics are approximated as unimodal, zero-mean, Gaussian noise

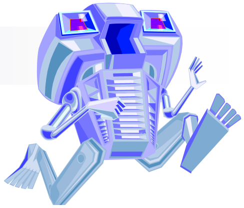
Kalman Filter Localization (5.6.3): Kalman Filter Theory

- Assume that 2 measurements were taken:
  - sonar at time *k*
  - laser at time *k+1*
- An estimate of robot position derived from
  - the sonar is q_1 with variance σ_1^2
  - the laser is q_2 with variance σ_2^2
- The 2 robot position estimates are:

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

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





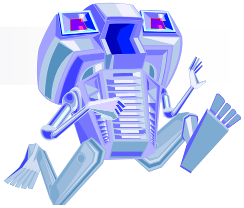
Kalman Filter Localization (5.6.3): Kalman Filter Theory

- How do we *fuse (combine)* these data to get the best estimate, $\hat{q}$
- Apply the weighted least-squares technique

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

- To find the minimum error set the derivative of S equal to zero

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



Kalman Filter Localization (5.6.3): Kalman Filter Theory

- Rearranging the equation, the estimate of the position in terms of the 2 measurements can be defined as

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

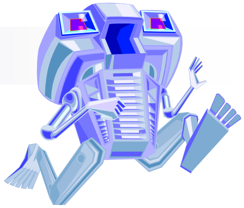
- In Kalman Filter notation,

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

where

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

Kalman Filter Localization (5.6.3): Dynamic estimation



- What if the robot moves between successive sensor measurements?
- the robot motion between *k* and *k + 1* is described by the velocity, *u*, and the noise, *w*

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

- If we know the robot’s variance at *k* is σ_k^2 and the variance of the motion is σ_w^2 then from *k* , the time when the measurement is taken yields

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

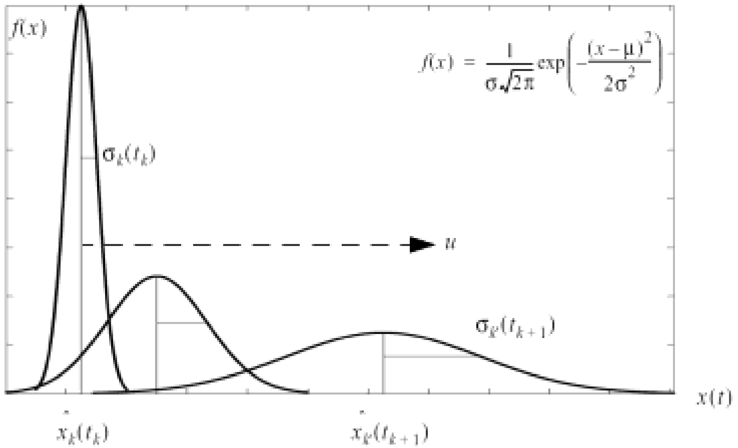
$$\sigma_{k'}^2 = \sigma_k^2 + \sigma_w^2[t_{k+1} - t_k]$$

Kalman Filter Localization (5.6.3): Dynamic estimation

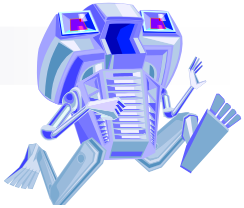
- x_k is the optimal prediction of the robot's position just as the measurement is taken at time *k + 1*
- It describes the growth of position error until a new measurement is taken
- The optimal estimate at time *k+1* is given by the last estimate at k and the estimate of the robot motion including the estimated movement errors

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

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



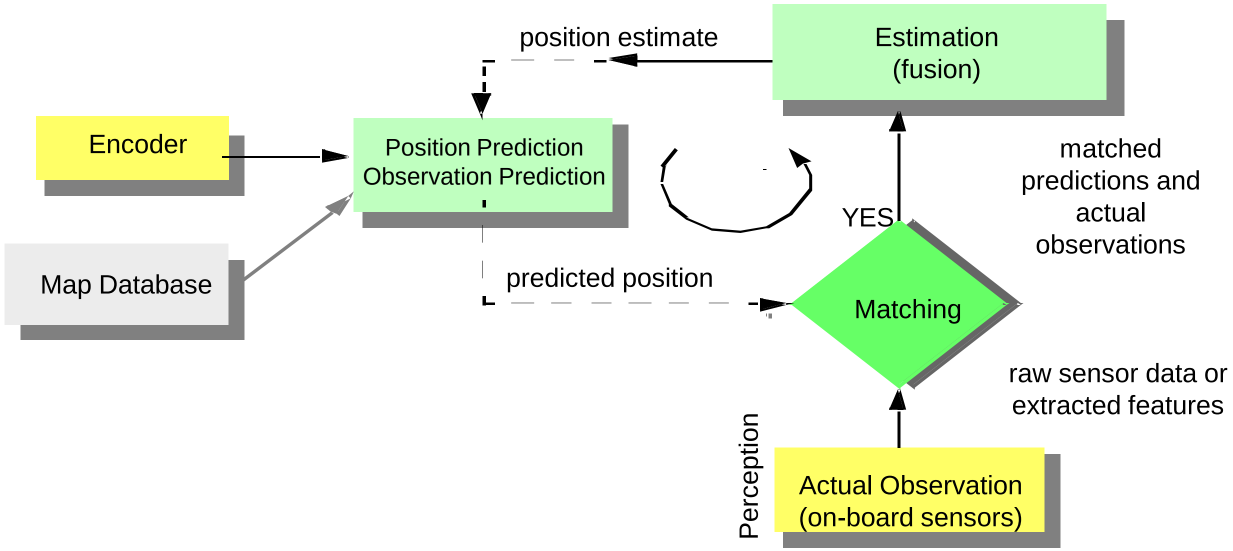
Kalman Filter Localization (5.6.3): Application to mobile robots

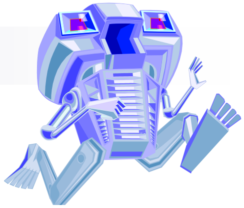


- The application of Kalman filters to localization requires posing the robot localization problem as a sensor fusion problem
- Recall that the basic probabilistic update of the robot belief statement can be segmented into 2 phases
  - *perception update*
  - *action update*
- The key difference between Markov and Kalman lies in the perception update process
  - the entire perception, the robot’s set of instantaneous sensor measurements, is used to update each possible robot position in the belief state individually
  - for Dervish, the perception was abstract being produced from a feature extraction mechanism
  - for Rhino, the perception consists of raw sensor readings
- For the Kalman filter, perception update is a multistep process

Kalman Filter Localization (5.6.3): Application to mobile robots

- In Kalman *perception update*, the robot's total sensory input is treated as a set of extracted features that each relate to objects in the environment
- The Kalman filter treats the whole belief state at once

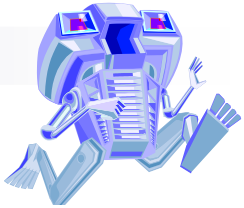




Kalman Filter Localization (5.6.3): Steps for Kalman filter localization

- action update or *position prediction*
  - Gaussian error model to the robot’s measured encoder travel
- *observation step*
  - robot collects actual senor data and extracts appropriate features
- *measurement prediction*
  - at the same time, based upon the robot’s predicted position in the map, the robot identifies the features that the robot expects to find and the positions of those features
- *matching*
  - the robot identifies the best pairings between the features actually extracted during observation and the expected features due to measurement prediction
- *estimation*
  - Kalman filter fuses the information provided by all of the matches to update the robot belief state

Kalman Filter Localization (5.6.3.2): Robot position prediction (Step 1)



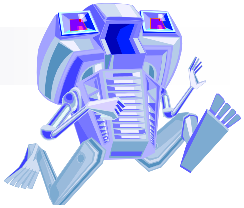
- The robot’s position at time step *k*+1 is predicted based on its old location (time step *k*) and its movement due to the control input *u*(*k*):

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

- Knowing the plant and error model, the variance associated with the prediction is

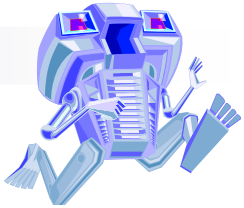
$$\Sigma_p(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u f^T$$

- This allows the prediction of the robot’s position and its uncertainty after a movement specified by the control input



Kalman Filter Localization (5.6.3.2): Observation (Step 2)

- To obtain the sensor measurements *$Z(k+1)$* from the robot's sensors at the new location at time *$k+1$*
- assume that the observation is the result of a feature extraction process executed on raw sensor data
- The observation usually consists of a set *n_o* of single observations *$z_j(k+1)$* extracted from the different sensors signals (e.g. raw data scans, or features such as lines, doors, 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).



Kalman Filter Localization (5.6.3.2): Measurement Prediction (Step 3)

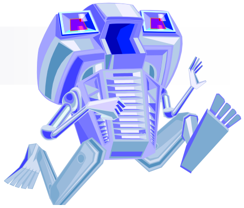
- Use the predicted robot position $\hat{p} = (k+1|k)$ and the map *M*(*k*) to generate multiple predicted feature observations z_t.
- Each predicted feature has to be transformed into the sensor frame

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

- Define the measurement prediction as the set containing all n_i predicted feature observations

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

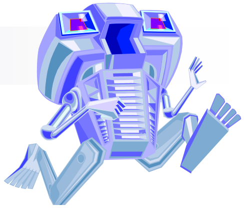
- The function h_i is mainly the coordinate transformation between the world frame and the sensor frame



Kalman Filter Localization (5.6.3.2): Matching (Step 4)

- identifies all of the single observations that match specific predicted features well enough to be used during the estimation process
- produce an 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 an corresponding observation is found we calculate the *innovation*
- *Innovation* is the measure of the difference between the predicted and observed measurements

$$
\begin{aligned}
v_{ij}(k+1) &= [z_j(k+1) - h_i(z_t, \hat{p}(k+1|k))] \\
&= \begin{bmatrix} \alpha_j \\ r_j \end{bmatrix} - \begin{bmatrix} {}^W\alpha_{t,i} - {}^W\hat{\theta}(k+1|k) \\ {}^Wr_{t,i} - ({}^W\hat{x}(k+1|k)\cos({}^W\alpha_{t,i}) + {}^W\hat{y}(k+1|k)\sin({}^W\alpha_{t,i})) \end{bmatrix}
\end{aligned}
$$



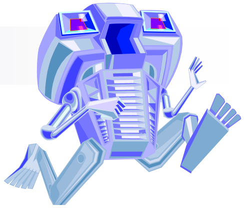
Kalman Filter Localization (5.6.3.2): Matching (Step 4)

- The *Innovation covariance* can be found by applying the error propagation law

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

- To determine the validity of the correspondence between measurement prediction and observation, a *validation gate* has to be specified. One definition is the *Mahalanobis distance*

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



Kalman Filter Localization (5.6.3.2): Applying the Kalman Filter (Step 5)

- Compute the best estimate of the robot’s position based on the position prediction and all the observations at time k+1
- The Kalman filter gain is used to update the robot’s position estimate

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

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

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

Kalman Filter Localization (5.6.3.3): Case Study (Line Feature Extraction)

- Pygmalion is a differential drive robot that uses a laser rangefinder as its primary sensor
- The environment representation is continuous and abstract
  - the map consists of a set of infinite lines describing the environment
- the belief state is a Gaussian distribution and uses the Kalman filter localization algorithm
- Assume that the sensor frame {S} is equal to the robot frame {R}
- Assume that if not specified all the vectors are represented in the coordinate system {W}

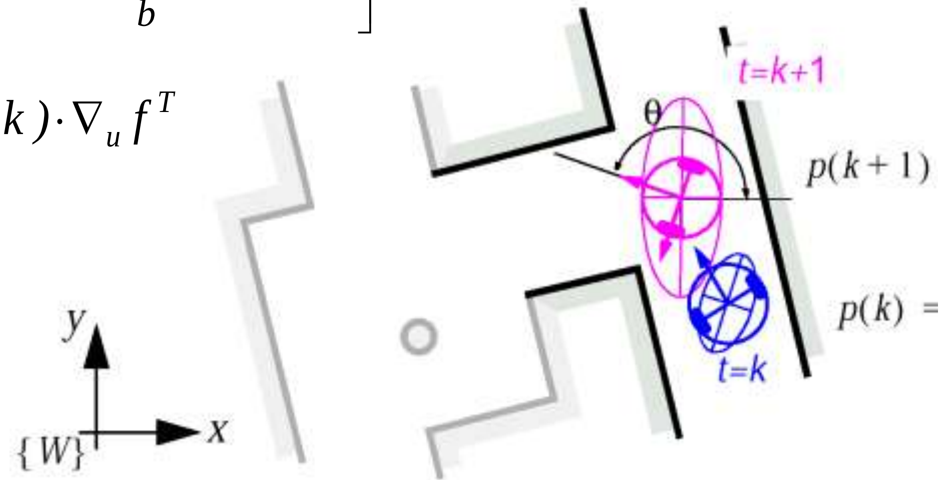
Kalman Filter Localization (5.6.3.3): Case Study (Robot position prediction)

- At the time increment *k* the robot is at position *p(k) = [x(k) y(k) θ(k)]*
- The control input *u(k)* drives the robot to the position *p(k+1)*
- The best position estimate is

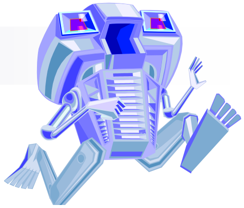
$$\hat{p}(k+1|k) = \hat{p}(k|k) + u(k) = \begin{bmatrix} \hat{x}(k) \\ \hat{y}(k) \\ \hat{\theta}(k) \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2}\cos(\theta + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r + \Delta s_l}{2}\sin(\theta + \frac{\Delta s_r - \Delta s_l}{2b})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

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

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



Kalman Filter Localization (5.6.3.3): Case Study (Observation)

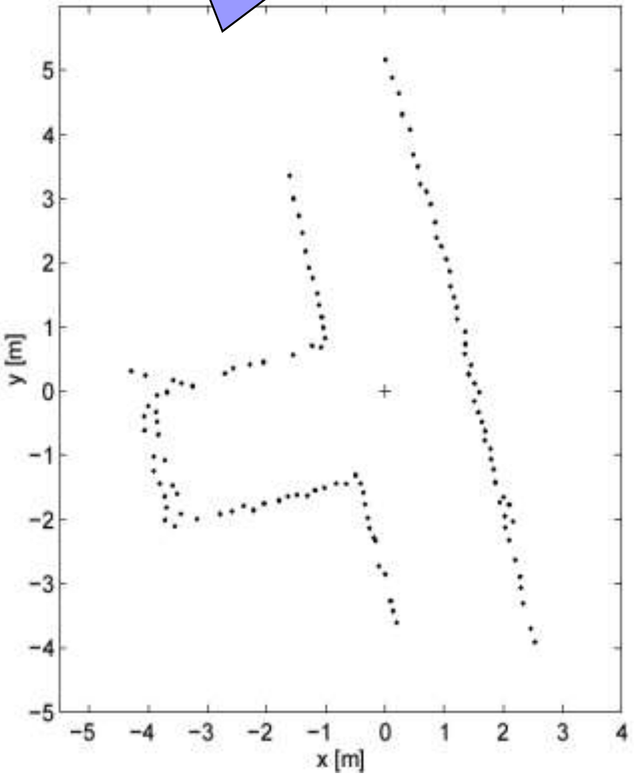


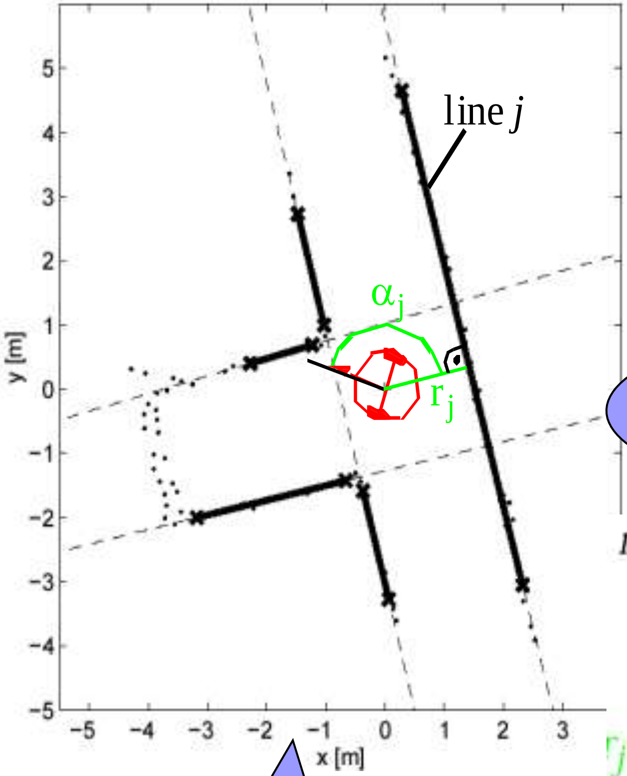
- for line-based localization, each single observation (i.e., a line feature) is extracted from the raw laser rangefinder data and consists of $\beta_{0,j}$, $\beta_{1,,j}$ or α_j, r_j
- $z_j(k + 1) = {}^{R}[\alpha_j \;\; r_j]^T$
- lines and uncertainties are extracted and n_0 observations leads to $2n_o$ line parameters

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

Kalman Filter Localization (5.6.3.3): Case Study (Observation)

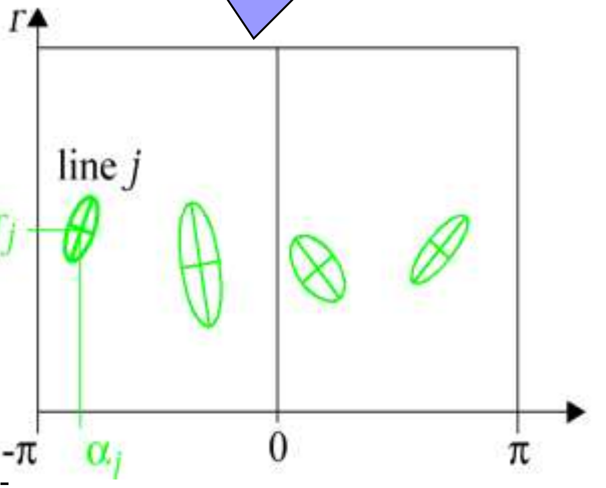
raw data from the laser scanner at time k+1, extracted lines





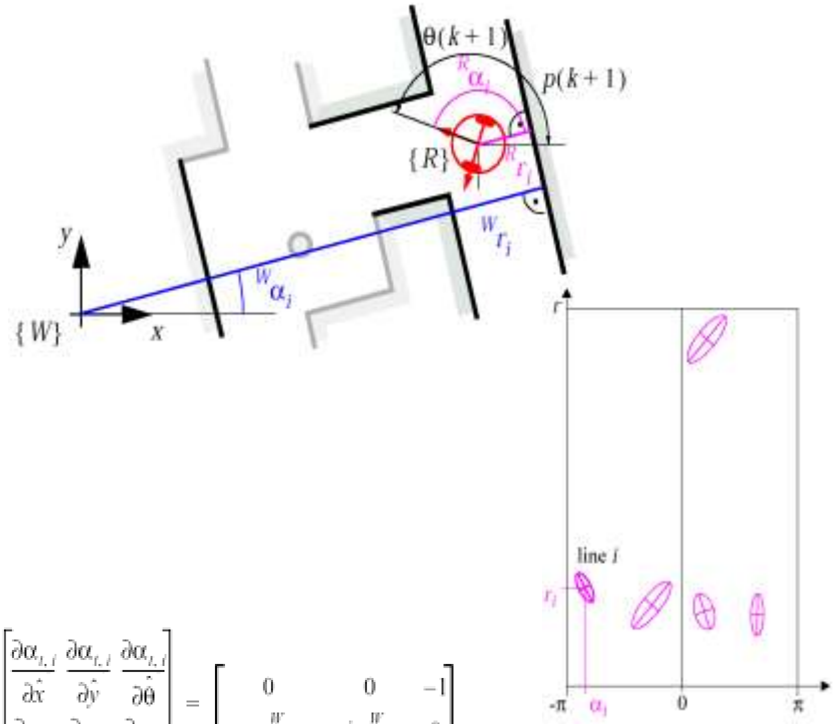
lines extracted from the raw data

the extracted lines uncertainties represented in the model space



Kalman Filter Localization (5.6.3.3): Case Study (Measurement prediction)

- Based on the stored map and the predicted robot position, the measurement predictions of expected features are generated
- To reduce required calculations, 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



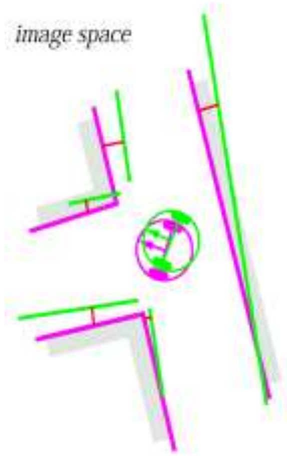
$$ {}^W z_{t,i} = {}^W\begin{bmatrix}\alpha_{t,i} \\ r_{t,i}\end{bmatrix} \rightarrow {}^R z_{t,i} = {}^R\begin{bmatrix}\alpha_{t,i} \\ r_{t,i}\end{bmatrix} $$

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

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

Kalman Filter Localization (5.6.3.3): Case Study (Matching)

- find a correspondence between predicted and observed features

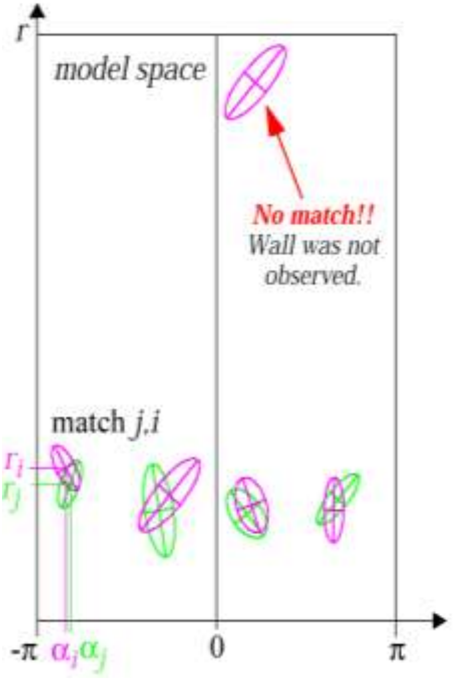


$$v_{ij}(k+1)\cdot\Sigma_{IN,\,ij}^{-1}(k+1)\cdot v_{ij}^{T}(k+1)\le g^{2}$$

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

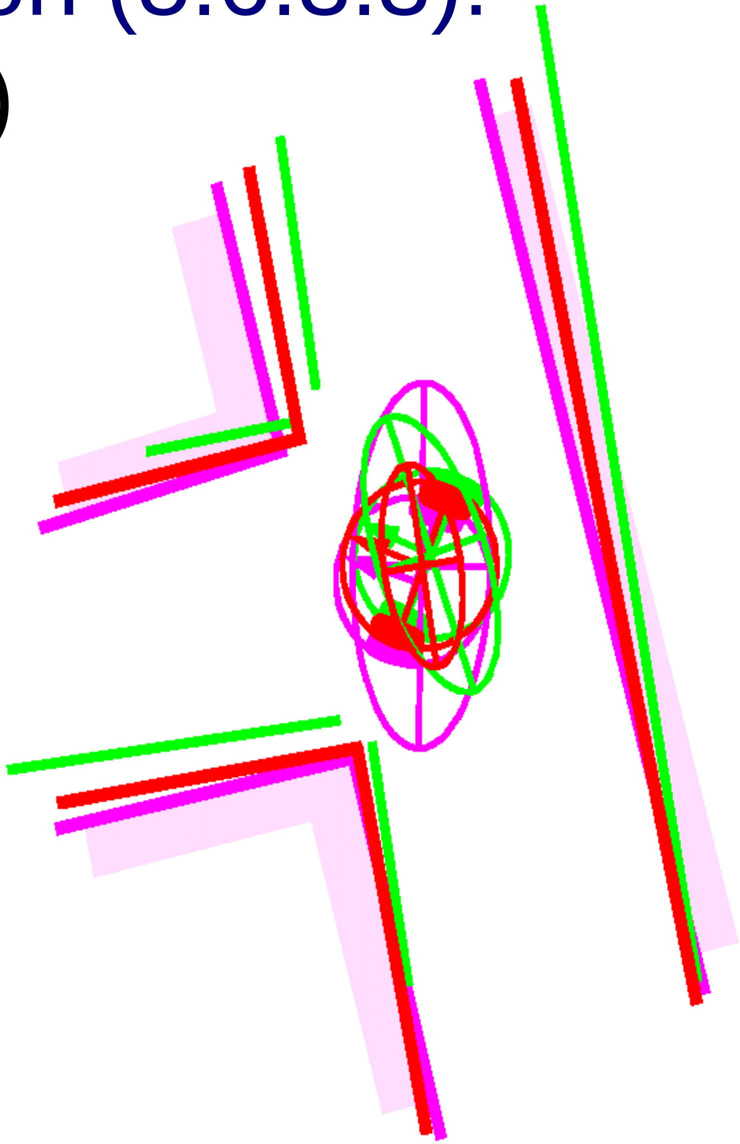
$$= \begin{bmatrix}\alpha_j\\ r_j\end{bmatrix} - \begin{bmatrix} {}^{W}\alpha_{t,\,i} - {}^{W}\hat{\theta}(k+1|k) \\ {}^{W}r_{t,\,i} - ({}^{W}\hat{x}(k+1|k)\cos({}^{W}\alpha_{t,\,i}) + {}^{W}\hat{y}(k+1|k)\sin({}^{W}\alpha_{t,\,i}))\end{bmatrix}$$

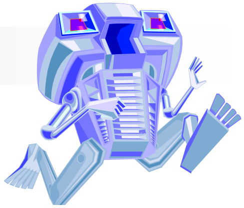
$$\Sigma_{IN,\,ij}(k+1) = \nabla h_i\cdot\Sigma_p(k+1|k)\cdot\nabla h_i^{T} + \Sigma_{R,\,i}(k+1)$$



Kalman Filter Localization (5.6.3.3): Case Study (Estimation)

- Kalman filter estimation of the new robot position
  - 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)
- this final pose estimate corresponds to the weighted sum of the
  - pose estimates of each matching pairing of observed and predicted features
  - robot position estimation based on odometry and observation positions



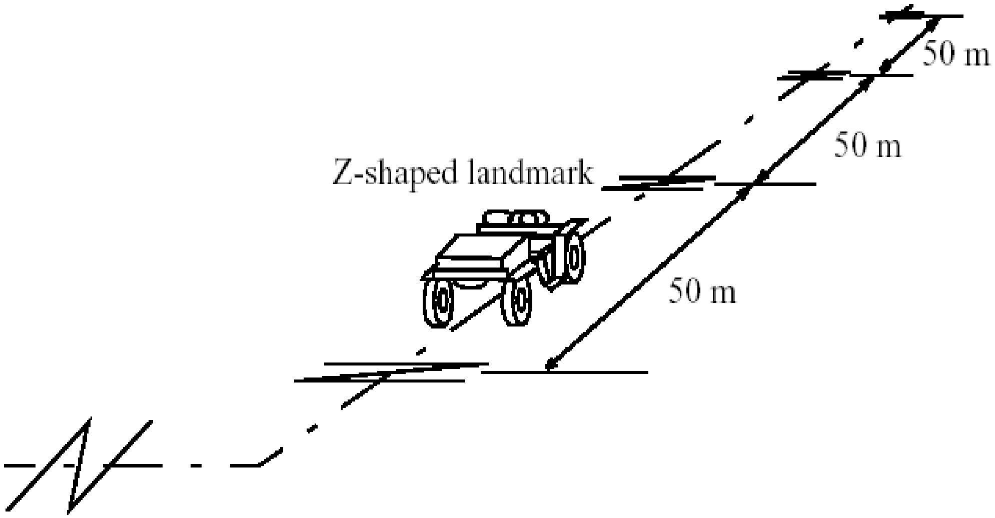


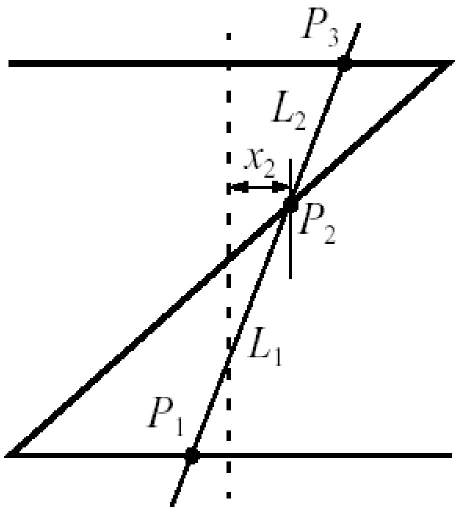
Other examples of Localization Systems: Landmark-based navigation (5.7)

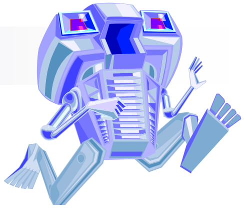
- Other methods use techniques that may modify the robot’s environment
- *landmarks* are passive objects in the environment that provide a high degree of localization accuracy when they are within the robot’s field of view
- the control system consists of 2 discrete phases
  - □ when the landmark is in view, the robot localizes frequently and accurately using action update and perception update to track its position without cumulative error
  - □ when the landmark is not in view, only the action update occurs and the robot accumulates position uncertainty until the next landmark enters the robot’s field of view

Other examples of Localization Systems (5.7): Landmark-based navigation

- the robot is *dead reckoning* from landmark zone to landmark zone
- the robot must consult its map carefully
- the shape of the landmarks may enable reliable and accurate pose estimation by the robot

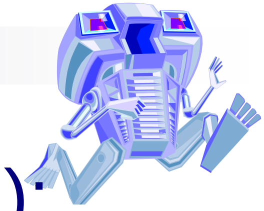






Other examples of Localization Systems (5.7): Landmark-based navigation

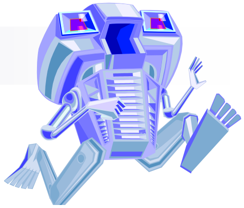
- One key advantage of landmark-based navigation is that a strong formal theory has been developed for this general system architecture
- the robot is *dead reckoning* from landmark zone to landmark zone
- the robot must consult its map carefully
- the shape of the landmarks may enable reliable and accurate pose estimation by the robot
- the disadvantage is that the environment must be modified and the landmarks are local, therefore a large number are required to cover a given area



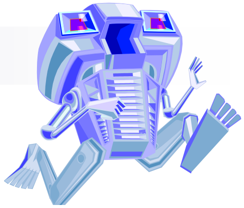
Other examples of Localization Systems (5.7.2): Globally unique localization

- The general assumption is that when the landmark is within the robot’s field of view, localization is essentially perfect
- To greatly improve robot localization, this assumption would be true no matter where the robot is located
- a look at the robot’s sensors would immediately identify its particular location, uniquely and repeatedly
- One type of globally unique localization is *mosaic-based localization* which takes advantage of the fine-grained floor texture using a CCD camera pointed at the floor
- Humans often have excellent *local positioning systems* in environments that are nonrepeating and well-known

Other examples of Localization Systems: Positioning Beacon Systems (5.7.3)

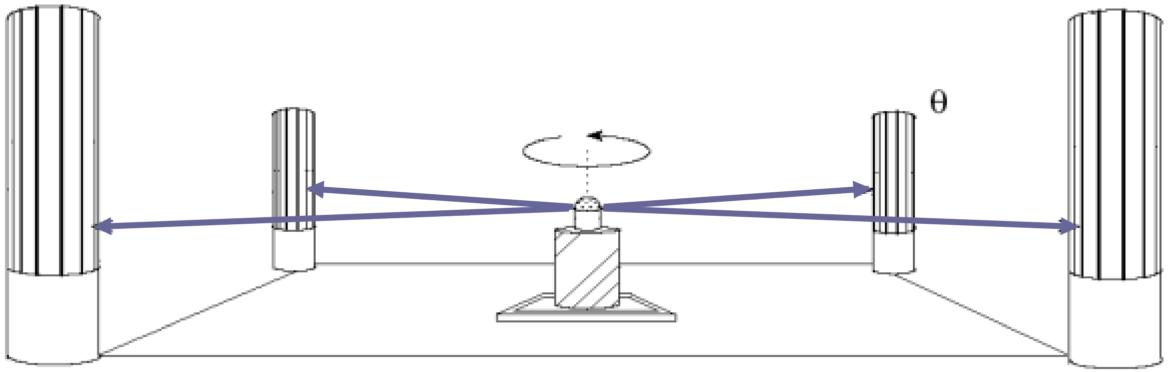


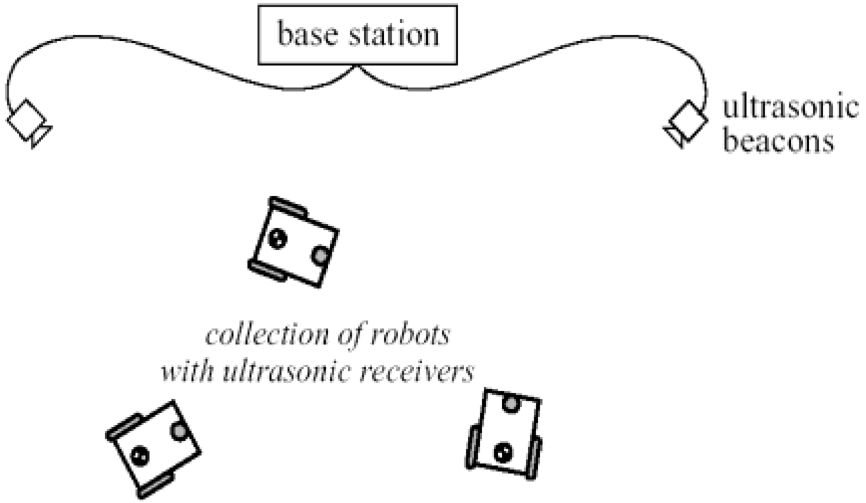
- one solution is to design and deploy an active beacon system specifically for the target environment
- Similar GPS, this method is preferred by industry and military applications to insure high reliability of localization
- The robots localize passively while the beacons are active
- any number of robots can simultaneously take advantage of a single beacon system
- the robots must know the positions of the 2 active ultrasonic beacons in the global coordinate frame in order to localize themselves
- one system with retroreflective markers can be easily detected by the robot based on their reflection of energy back to the robot when it has 3 beacons in sight simultaneously
- a robot with encoders can localize over time and does not need to measure all three beacons at the same instant



Other examples of Localization Systems:

Positioning Beacon Systems: Triangulation (5.7.3)





Other examples of Localization Systems:
Positioning Beacon Systems: Bar-Code

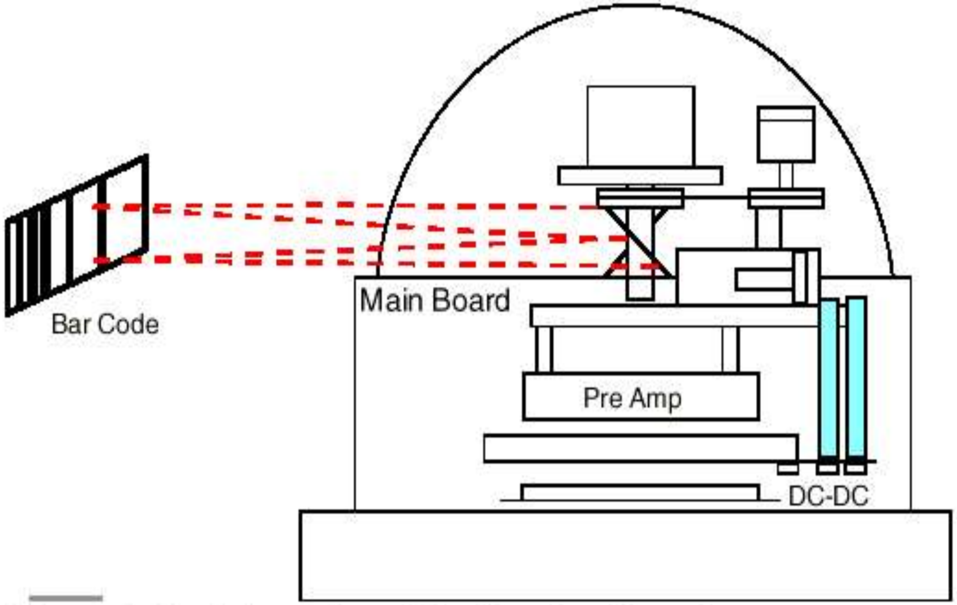


Figure 6.14: Schematics of the Denning Branch International Robotics *LaserNav* laser-based scanning beacon system. (Courtesy of Denning Branch International Robotics.)

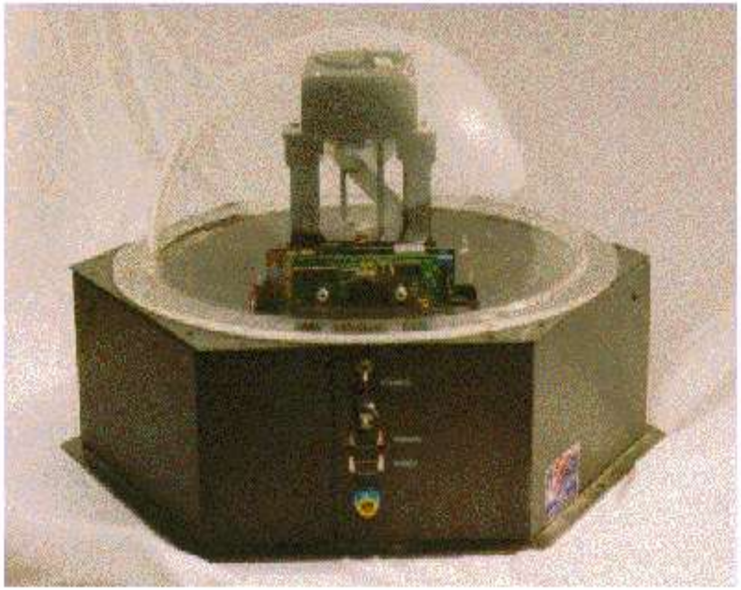
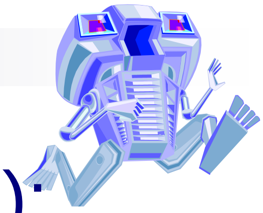
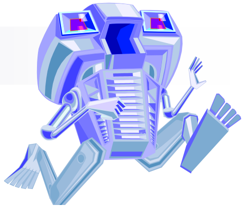


Figure 6.15: Denning Branch International Robotics (DBIR) can see *active targets* at up to 183 meters (600 ft) away. It can identify up to 32 active or passive targets. (Courtesy of Denning Branch International Robotics.)



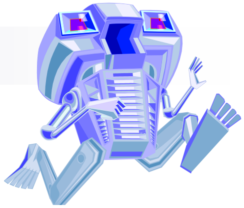
Other examples of Localization Systems (5.7.4): Route-based localization

- More reliable than beacon-based systems are *route-based localization*
- the route of the robot is explicitly marked and it can determine its position relative to the specific path it is allowed to travel
- this effectively creates a railway system, but more flexible (i.e. ultraviolet-reflective, optically transparent paint, guidewire)
- *Unmanned guide vehicles* use this technique but may deviate from their route to avoid obstacles
- this robot is much more inflexible and changes to robot behavior and the environment require significant engineering and time



Autonomous Map Building (5.8)

- A robot that localizes successfully has the right sensors for detecting the environment and the robot ought to build its own map
  - starting from an arbitrary initial point,
  - a mobile robot should be able to autonomously explore the environment with its sensors,
  - gain knowledge about it,
  - interpret the scene,
  - build an appropriate map
  - and localize itself relative to this map
- *Simultaneous Localization and Mapping (SLAM)* is one of the most difficult problems specific to mobile robot systems
  - SLAM involves the autonomous creation and modification of an environment map
  - the robot must explore its environment and build the map
  - the robot must also move and localize to explore the environment
  - the difficulty is based upon the interaction between the position updates and it localizes and the mapping actions

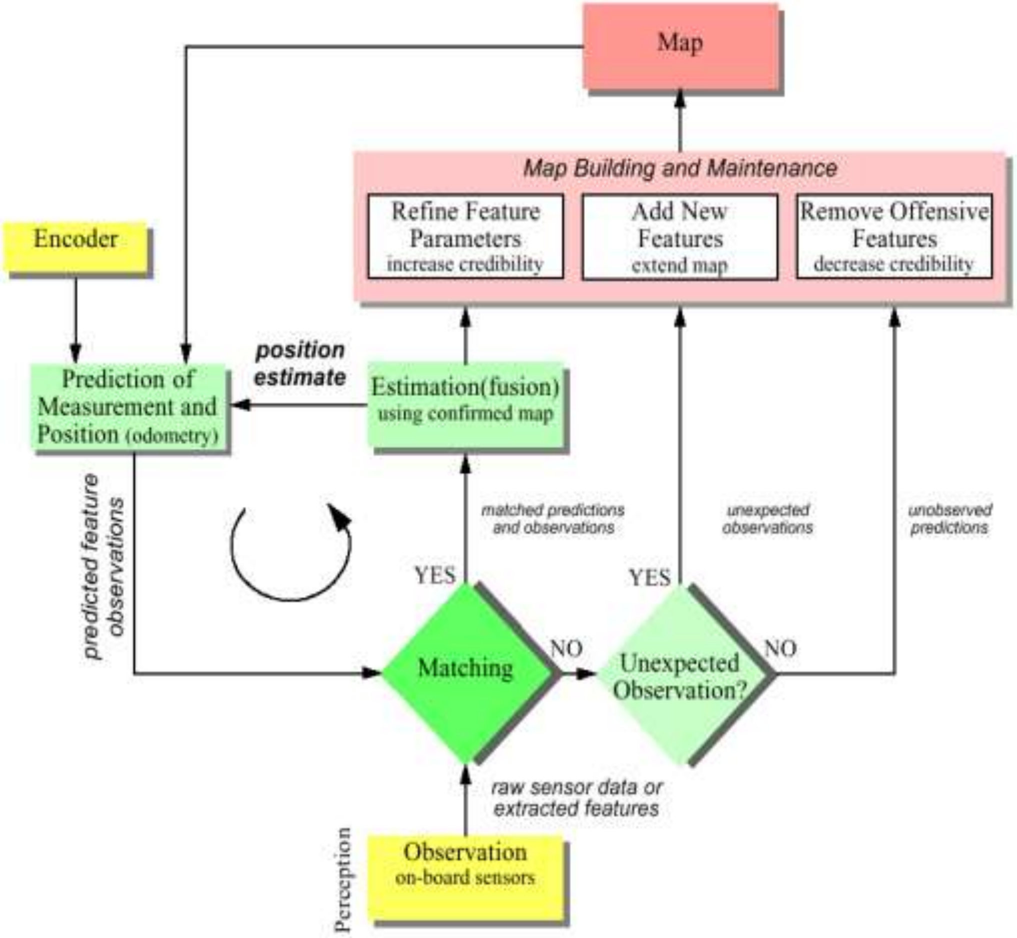


Autonomous Map Building (5.8): SLAM

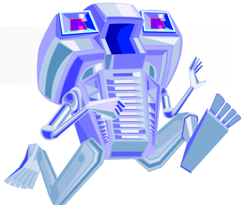
- SLAM is difficult based upon the interaction between the robot’s position updates as it localizes and the mapping actions
- if the robot updates its position based on an observation of an imprecisely known feature, the results position estimate becomes correlated with the feature location estimate
- the map becomes correlated with the position estimate if an observation taken from the imprecisely known position is used to update or add a feature to the map
- for localization, the robot needs to know where the features are but for map building, the robot needs to know where it is on the map
- the complete and optimal solution is to consider correlations between *position* and *feature location estimation*
- Cross-correlated maps are called *stochastic maps*

Autonomous Map Building (5.8.1): The stochastic map technique

- this general schematic incorporates map building and maintenance into the standard localization loop
- the added arcs represent the additional flow of information when there is an imperfect mach between observations and measurement predictions
- unexpected observations will effect the creation of new features in the map
- unobserved measurement predictions will effect the removal of features from the map

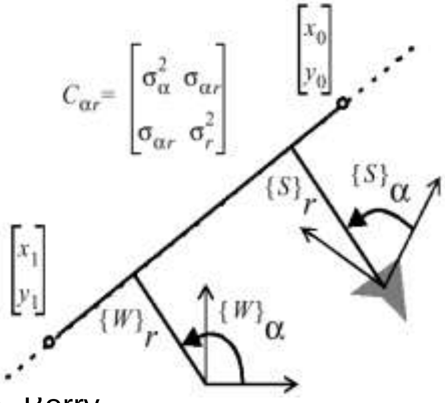


Autonomous Map Building (5.8.1):
The stochastic map technique

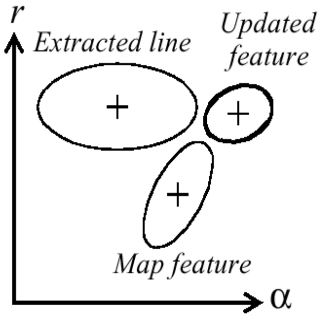


- each prediction or observation has an unknown exact value and is represented by a distribution
- the uncertainties of all of these quantities must be considered throughout the process
- each feature has varying degrees of probability
- the new map *M* with a set *n* of probabilistic feature locations z_t each with the covariance matrix and an associated credibility factor c_t between 0 and 1 quantifying the belief in the existence of the feature in the environment

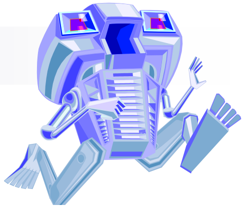
$$M = \{\hat{z}_t, \Sigma_t, c_t \,|\, (1 \le t \le n)\}$$



$C_{\alpha r} = \begin{bmatrix} \sigma_\alpha^2 & \sigma_{\alpha r} \\ \sigma_{\alpha r} & \sigma_r^2 \end{bmatrix}$
$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$
$\{S\}r$
$\{S\}\alpha$
$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$
$\{W\}r$
$\{W\}\alpha$



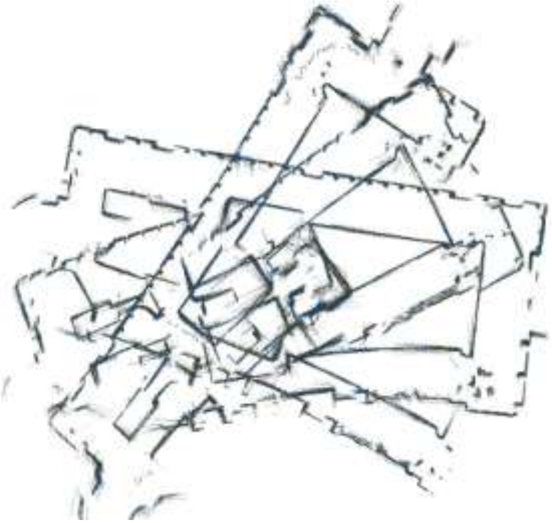
Autonomous Map Building (5.8): Stochastic Map Technique

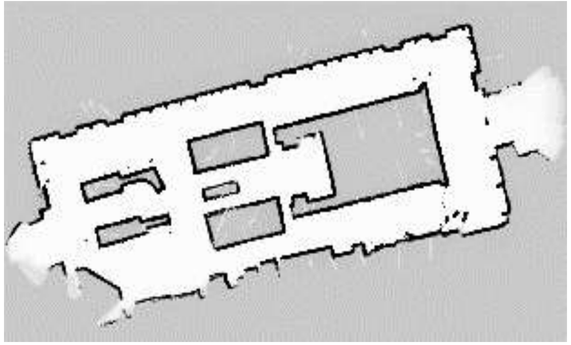


- in contrast to the map used for Kalman filter localization, the map M is not assumed to be precisely known because it will be created by an uncertain robot over time
- the matching step has 3 outcomes in regard to measurement predictions and observations
  - matched prediction and observation
  - unexpected observations
  - unobserved predictions
- localization or position update proceeds as before but the map is also updated now
- *the credibility factor, c_t* governs the likelihood that the mapped feature is indeed in the environment
- in map-building the feature positions and the robot's position are strongly correlated and this forces the use of a *stochastic map*, in which all cross-correlations are updated in each cycle
- this approach requires every value in the map to depend on every other value

Other mapping techniques (5.8.2.1) : Cyclic Environments

- in automatic mapping how do you correctly map an environment with one or more loops or cycles?
- Small local error accumulate to arbitrary large global errors!
- This is usually irrelevant for navigation
- However, when closing loops, global error does matter
- 2 features that solve the cyclic detection problem
  - *submaps* treated as a single sensor during the robot's position update
  - topological representation associated with the set of metric submaps





Other mapping techniques (5.8.2.2) : Dynamic Environments

- in dynamic environments, automatic mapping should capture the salient objects detected by its sensors
- the robot should have the flexibility to modify its maps as the positions of the salient objects change
- Dynamical changes require continuous mapping
- continuous mapping is a direct outgrowth of successful strategies for automatic mapping of unfamiliar environments
- If extraction of high-level features would be possible, the mapping in dynamic environments would become significantly more straightforward.
  - □ e.g. difference between human and wall
  - □ Environment modeling is a key factor for robustness