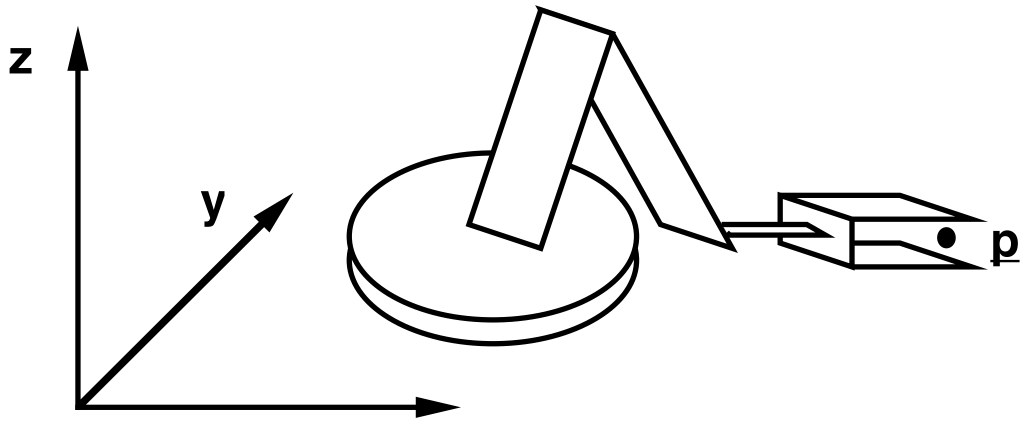
Additional Slides Robotics

Kinematics

Robot Localization

Geometry in Robotics

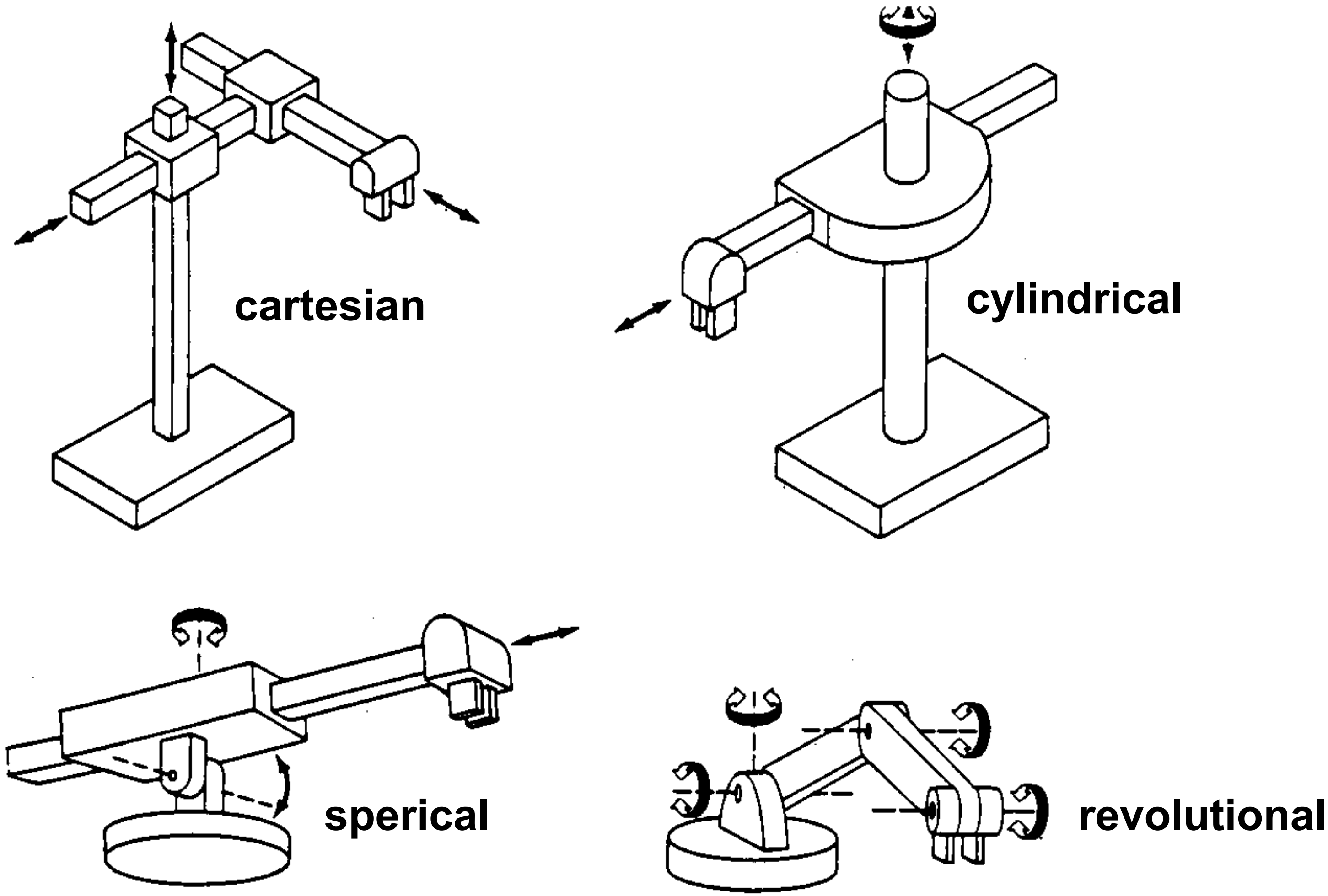
To carry out actions in its environment, a robot must understand the geometry of its body and its actuators relative to the geometry of the environment. This is called <u>kinematics</u>.



<u>Direct kinematics</u>:
Joint positions -> grasper position

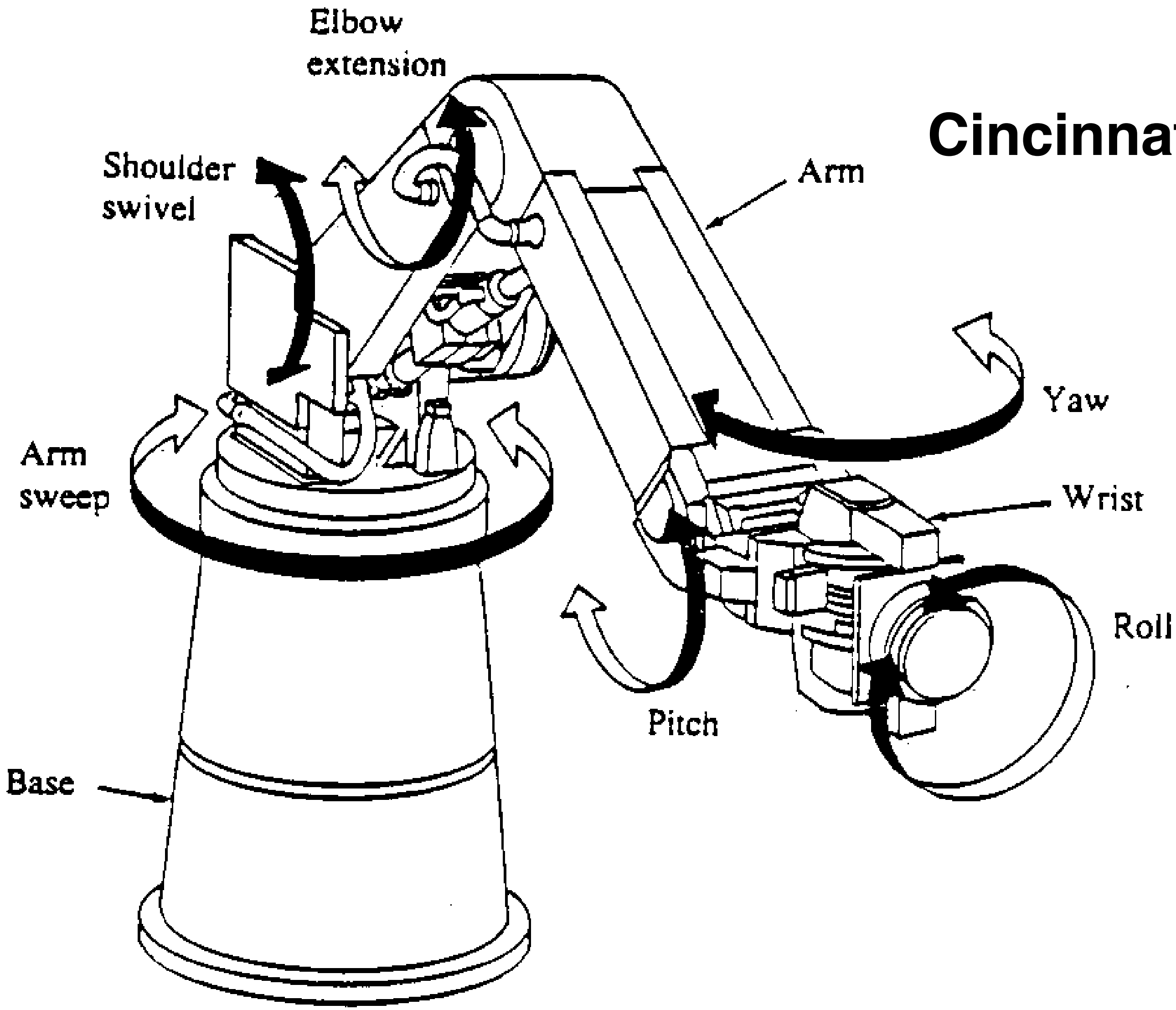
<u>Inverse kinematics</u>:
Grasper position -> joint positions

Simple Workspaces



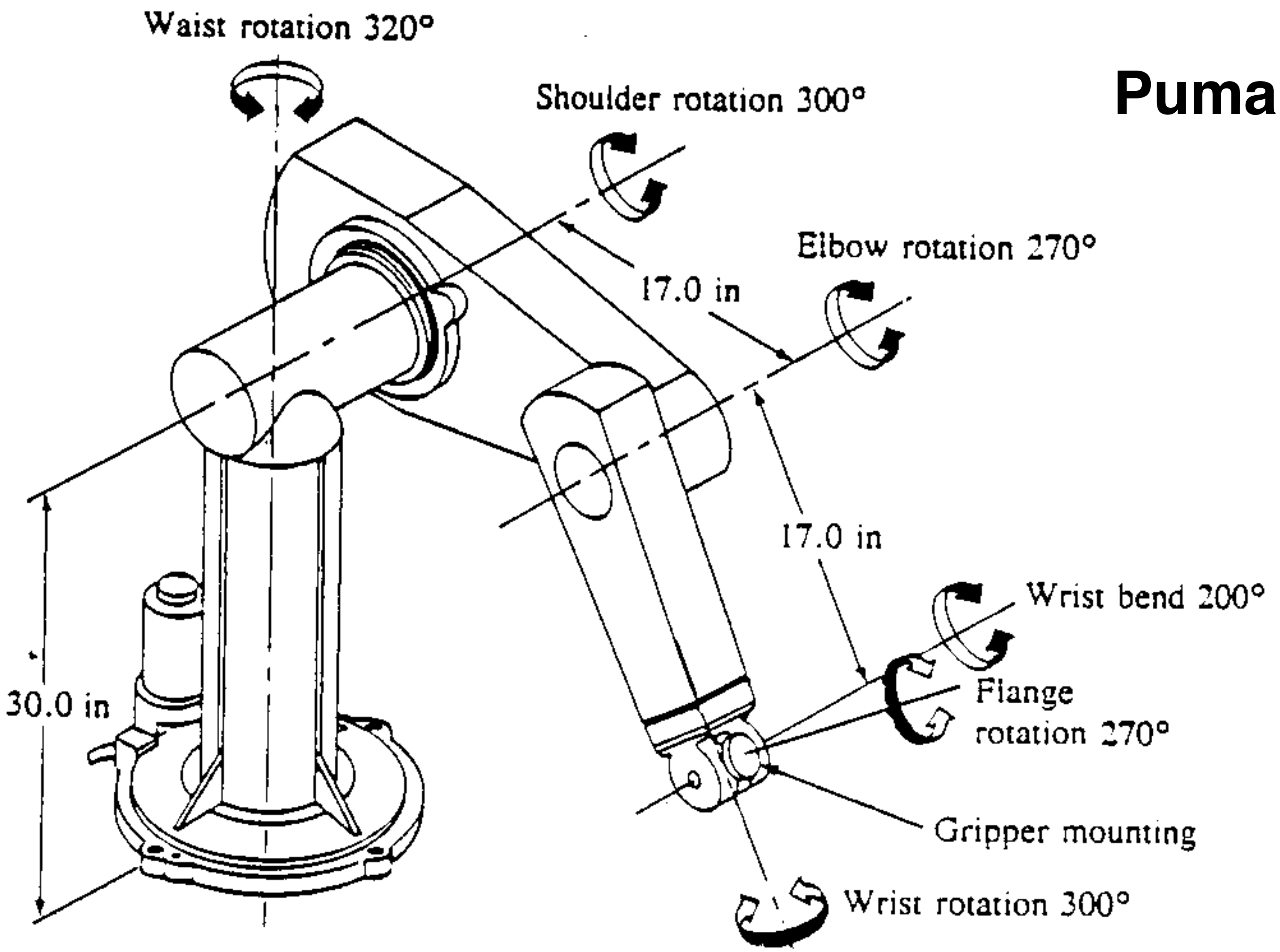
Workspace of an Industrial Robot (1)

Cincinnati Milacron T3



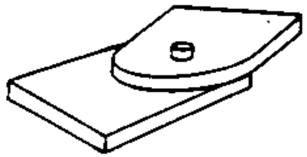
Workspace of an Industrial Robot (2)

Puma Arm

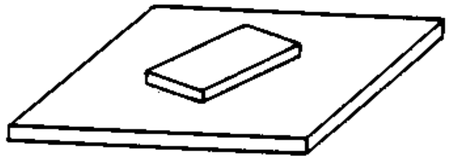


Six Possible Joint Types

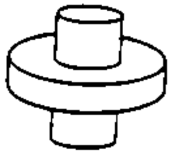
rotational joint



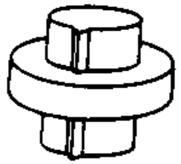
planar joint



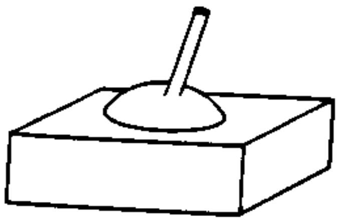
cylindrical joint



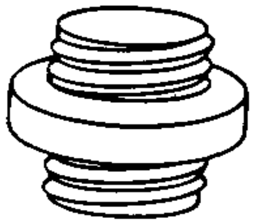
prismatic joint



spherical joint



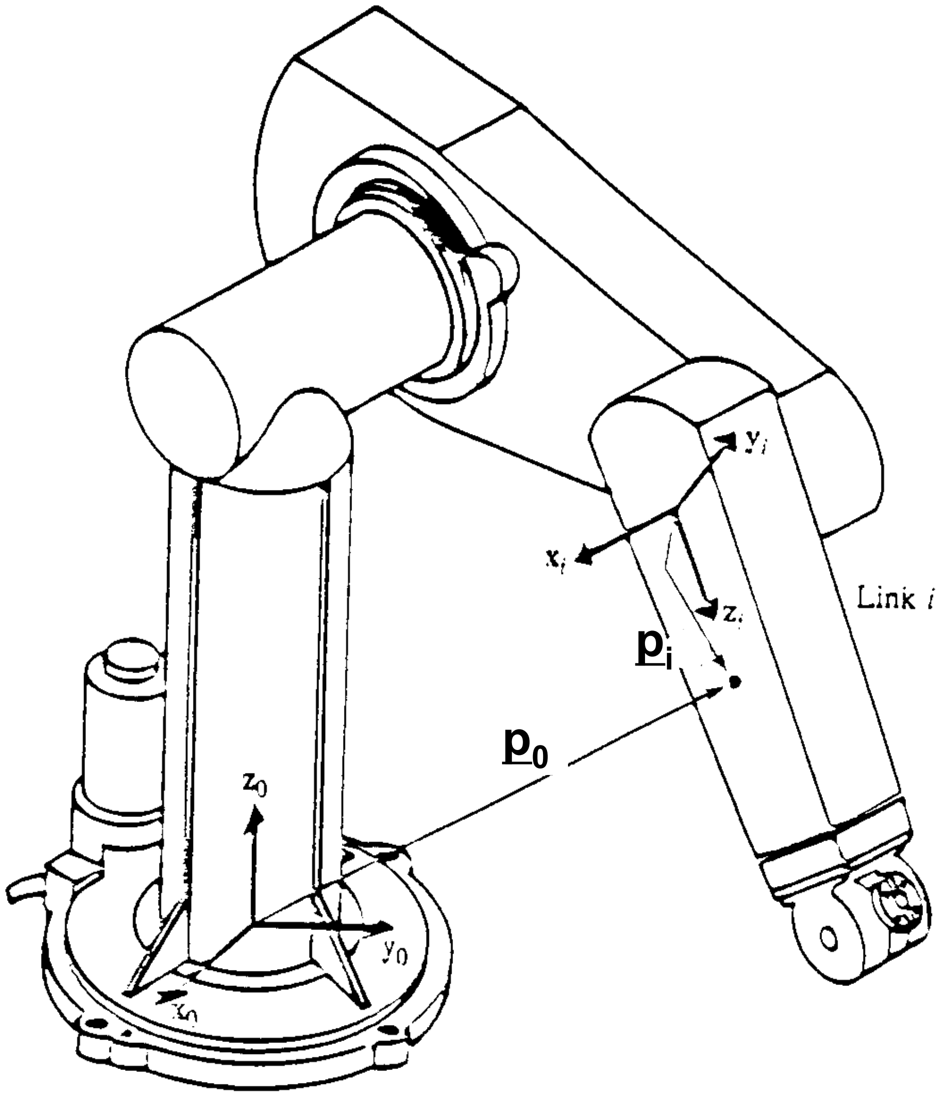
helical joint



Link Coordinate Systems

A link coordinate system is firmly attached to a link of the robot.

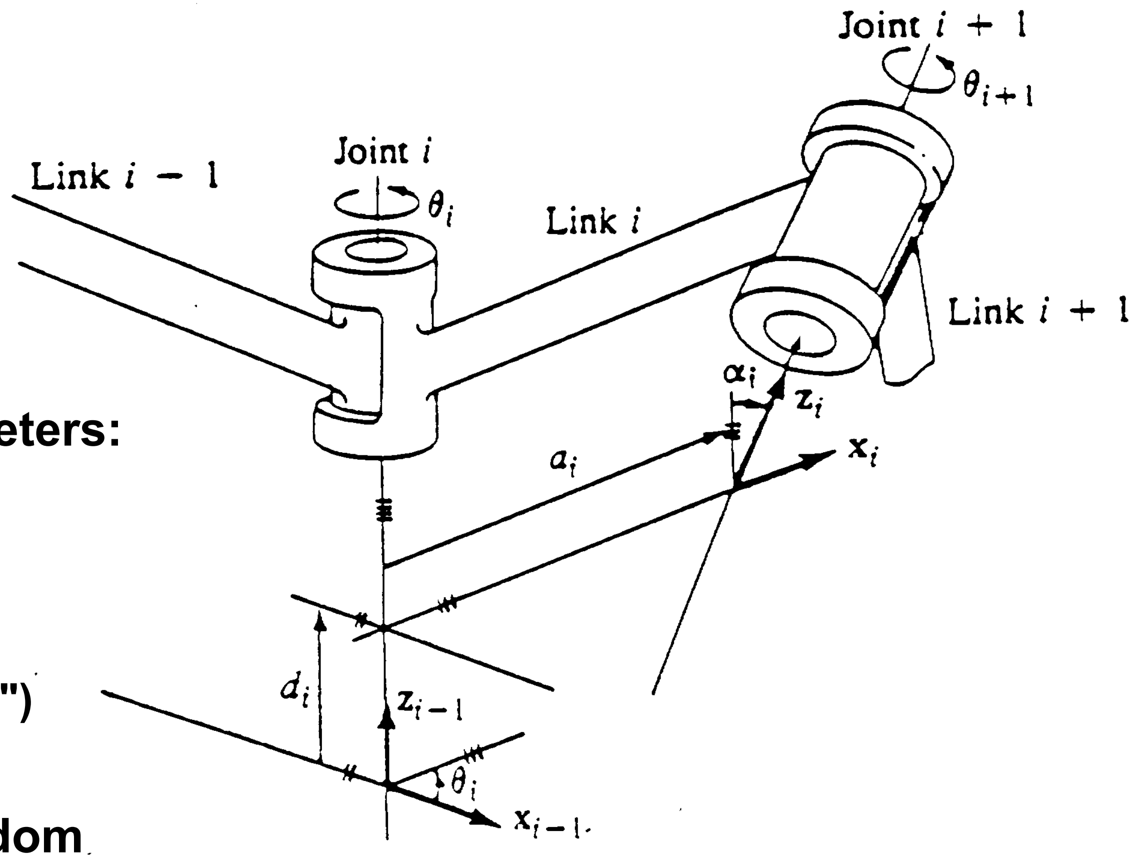
A point p can be represented in any link coordinate system.



Canonical Link Coordinates

Denavit-Hartenberg representation

z_{i-1}, z_i, z_{i+1} joint axes (translation or rotation)



Relative orientation and rotation of 2 consecutive links defined by 4 parameters:

- a_i smallest distance between z_i and z_{i-1}
- d_i distance between x_i and x_{i-1} along z_{i-1}
- α_i angle between z_i and z_{i-1}
- θ_i angle between x_i and x_{i-1} ("joint angle")

Typically, a joint has 1 degree of freedom (DoF), so 1 parameter is variable and 3 parameters are fixed.

Link Coordinate Transforms (1)

A point $\underline{p}_i$ in the ith link coordinate system can be expressed as $\underline{p}_{i-1}$ in the (i-1)th link coordinate system:

$$\underline{p}_{i-1} = \begin{bmatrix} \text{rotated } -\theta_i \\ \text{about } z_{i-1} \end{bmatrix} \begin{bmatrix} \text{translated } d_i \\ \text{along } z_{i-1} \end{bmatrix} \begin{bmatrix} \text{translated } a_i \\ \text{along } x_{i-1} \end{bmatrix} \begin{bmatrix} \text{rotated } \alpha_\iota \\ \text{about } x_i \end{bmatrix} \underline{p}_i$$

Rotation about x-axis:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}$$

Homogeneous coordinates: $A_\alpha = \left[\begin{array}{ccc|c} & & & 0 \\ & R & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$

Translation:

$$\underline{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Homogeneous coordinates: $A_t = \left[\begin{array}{ccc|c} 1 & 0 & 0 & \\ 0 & 1 & 0 & \underline{t} \\ 0 & 0 & 1 & \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$

Link Coordinate Transforms (2)

$\underline{p}_{i-1} = A_{-\theta} A_d \; A_a \; A_{-\alpha} \; \underline{p}_i$ in homogeneous coordinates

$= A_i \; \underline{p}_i$

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \text{sinai} \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\text{sinai} \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin \alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\underline{p}_0 = A_1 \ldots A_N \underline{p}_N$ world coordinates for point $\underline{p}_N$ in grasper coordinates

Example application: Test whether grasper tip $\underline{p}_N$ does not collide with obstacles

Homogeneous Coordinates

4D notation for 3D coordinates which allows to express nonlinear 3D transformations as linear 4D transformations.

Normal: $\underline{v}' = R\,(\underline{v} - \underline{v}_0)$ **rotation + translation**

Homogeneous coordinates: $\underline{\mathit{v}}' = \mathit{A}\,\underline{\mathit{v}}$ ***(note italics for homogeneous coordinates)***

$$A = R\,T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transition to homogeneous coordinates:

$\underline{v}^T = [x\ y\ z] \Rightarrow \underline{\mathit{v}}^T = [wx\ wy\ wz\ w]$ $w \neq 0$ **is arbitrary constant**

Return to normal coordinates:

1. **Divide components 1- 3 by 4th component**
2. **Omit 4th component**

Inverse Kinematics

Given:

T grasper position and orientation

Wanted:

A_i, i = 1 .. N joint positions such that $\prod_{i=1..N} A_i = T$

$\Rightarrow$ **12 nonlinear equations for N unknowns**

- **N ≥ 6 joint variable required for given position (3 degrees of freedom) and given orientation (3 DoF)**
- **Nonlinear equation, no guarantee for unique solutions**
- **Systematic solutions possible but not always practicable (precision, effort)**
- **Simple solutions for special manipulator geometry**

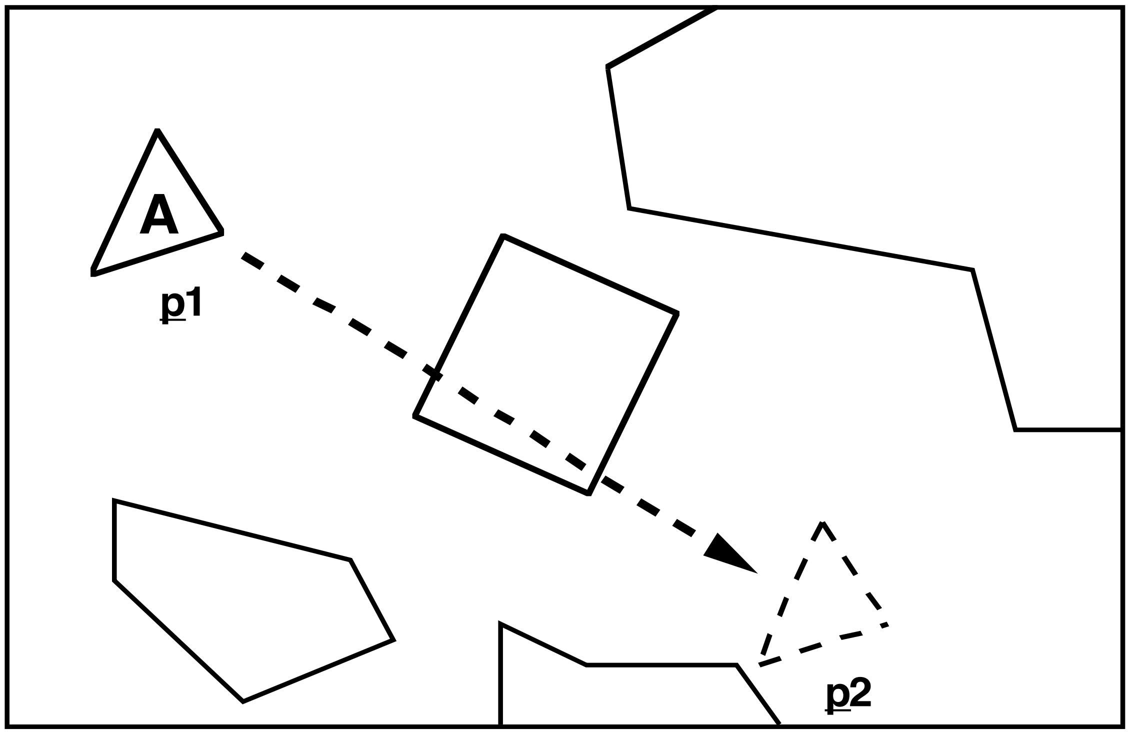
Path Planning (1)

How to move an object through an obstacle-crowded space from one point to another?

"Collision avoidance", "obstacle avoidance", "piano-movers problem"

<u>Example</u>:

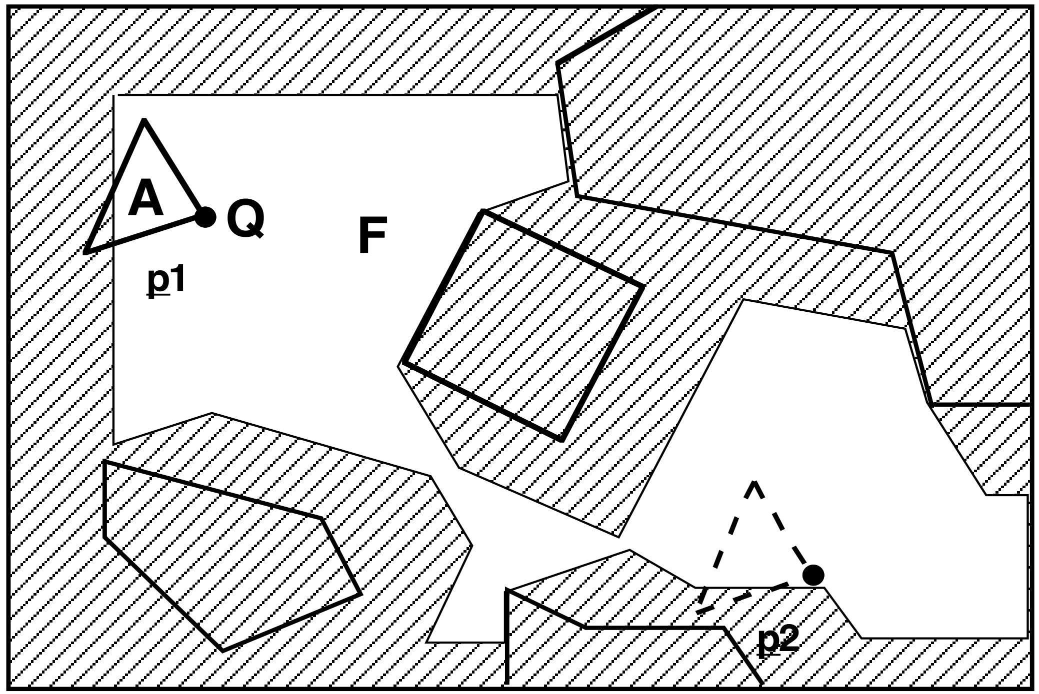
Move A from <u>p</u>1 to <u>p</u>2



Path Planning (2)

Basic idea:

1) **Determine free-space for reference point of A**
   - choose reference point
   - determine enlarged obstacles
   - examine manipulator workspace



2) **Search path for point object**
   - decompose freespace into free, occupied and mixed cells
   - search path through free cells, decompose mixed cells recursively

Configuration Space

- **Transformation of cartesian freespace coordinates into joint positions (C-space)**
- **C-space has 1 dimension for each DoF of the manipulator**
- **Mobility of manipulator determines boundaries of C-freespace**
- **Cartesian obstacles are transformed into C-space obstacles**
- **Path finding in C-space**

Problem:

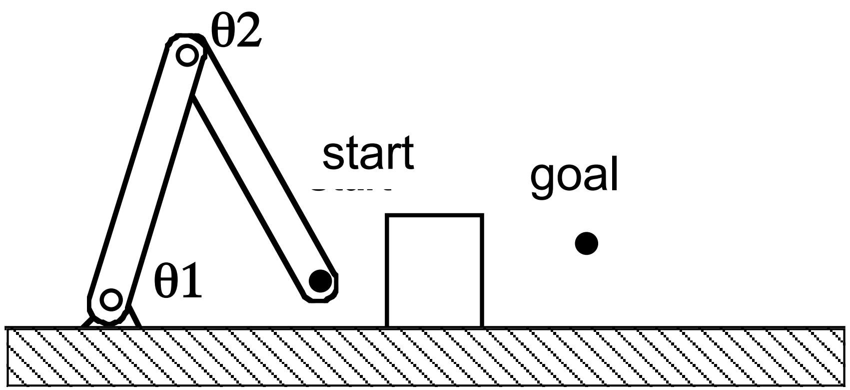
Transformation of cartesian coordinates into C-space requires inverse Kinematics

Path finding:

Which sequence of joint positions brings reference point of A from start to goal?

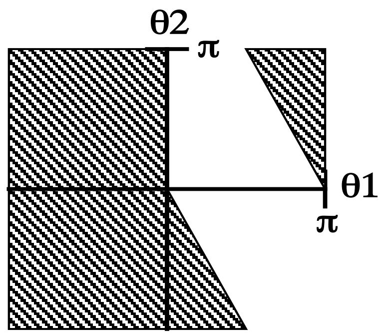
Example: 2-Joint Manipulator

Cartesian Space

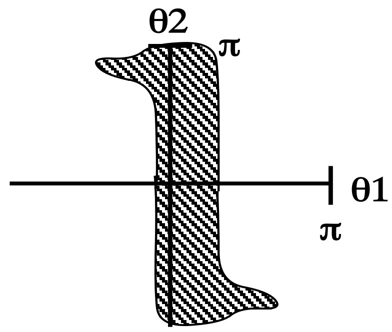


Configuration Space

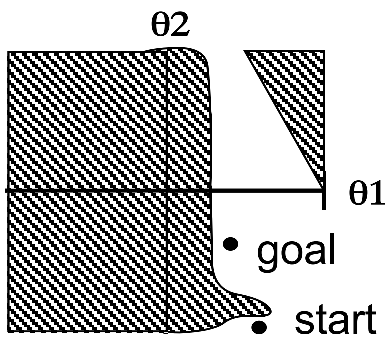
ground plane



box

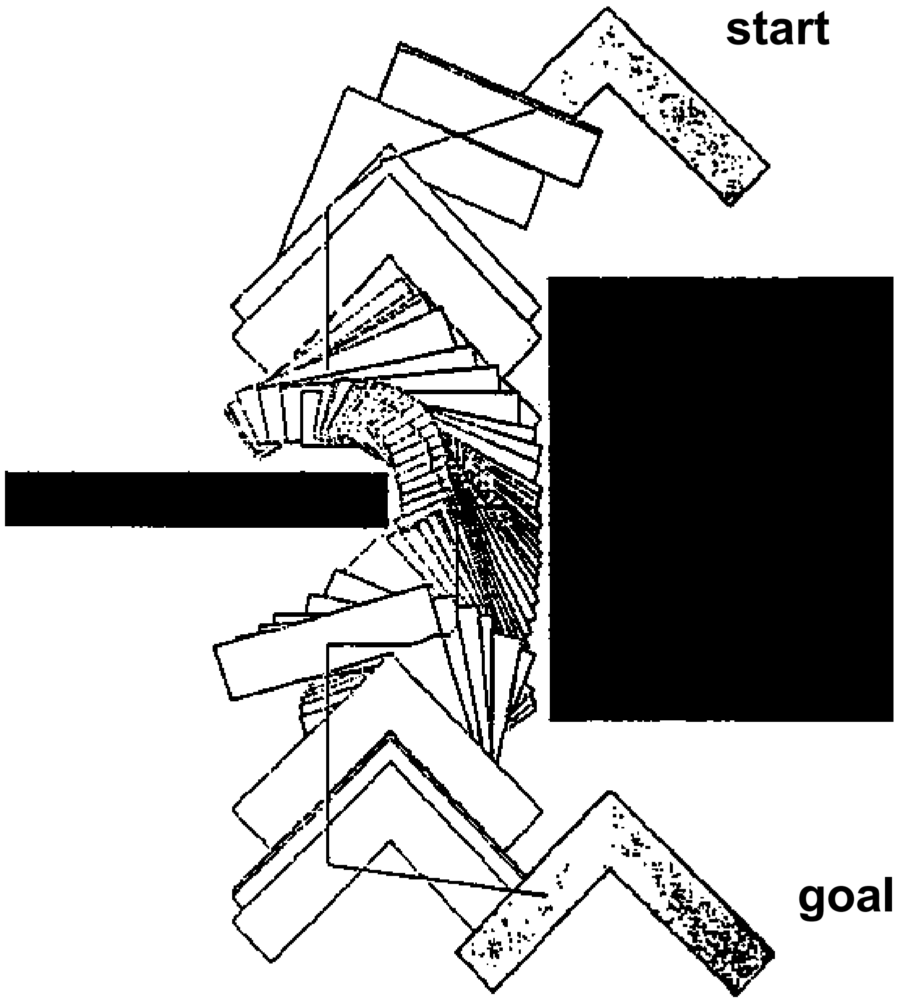


superposition



2D Path-Planning with Rotation

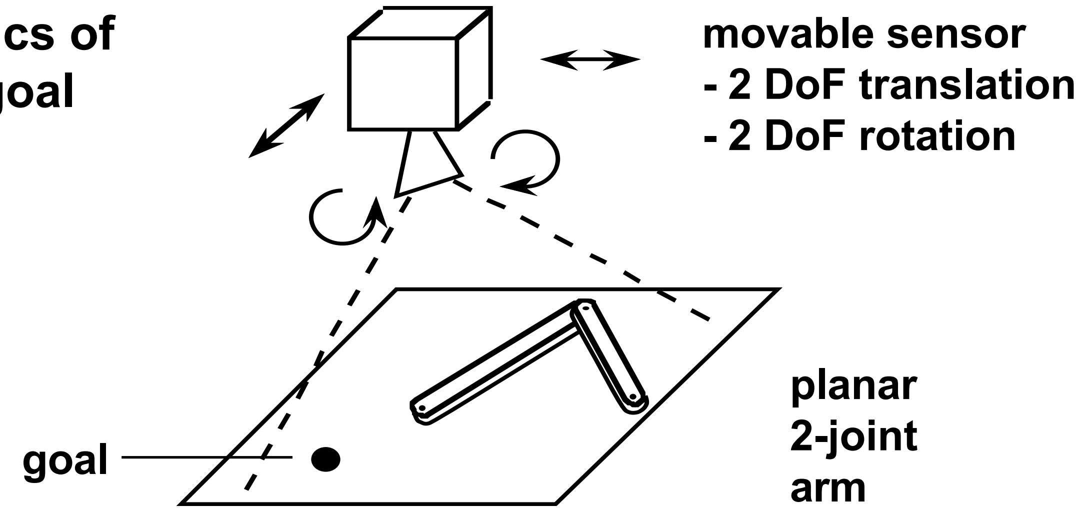
Example of Lozano-Perez:

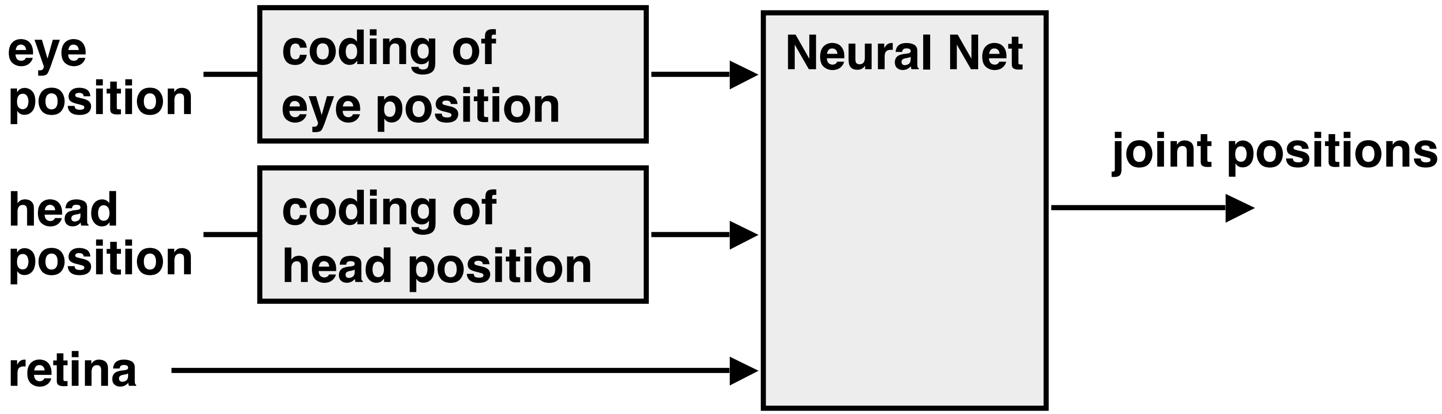


Learning Hand-Eye Coordination (1)

Pabon / Gossard: *Connectionist Networks for Learning Coordinated Motion in Autonomous Systems* AAAI-88

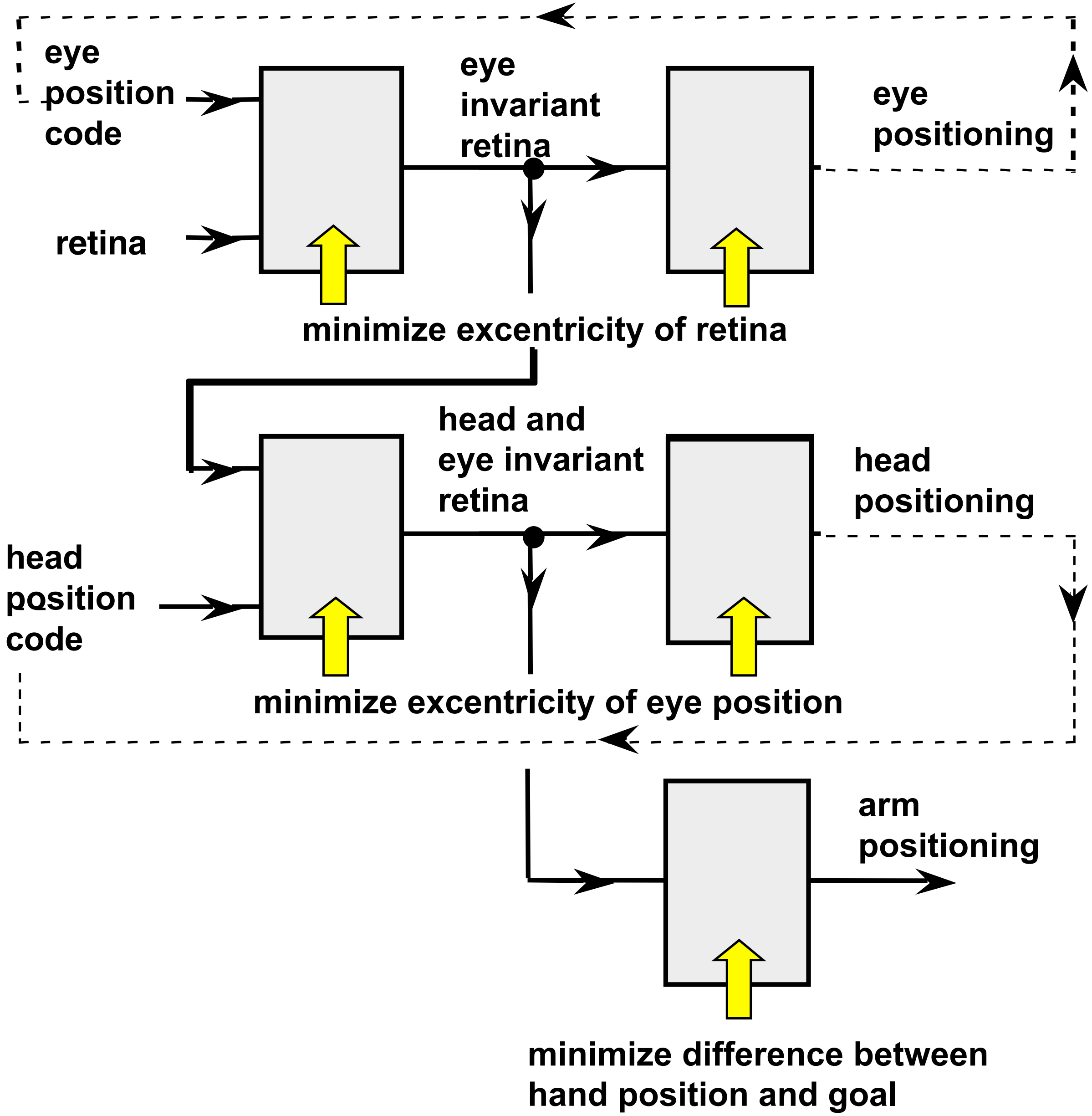
Learning the inverse kinematics of a 2-joint arm by observing a goal by a movable sensor





Learning Hand-Eye Coordination (2)

Each building block is a 2-layer feed-forward network

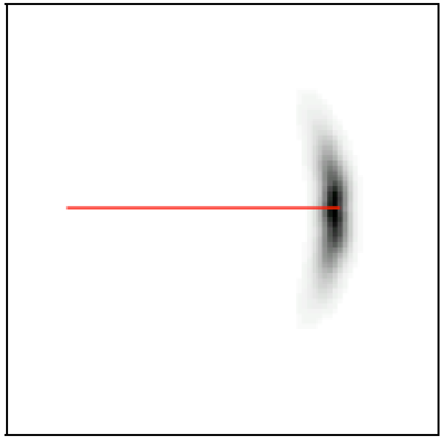


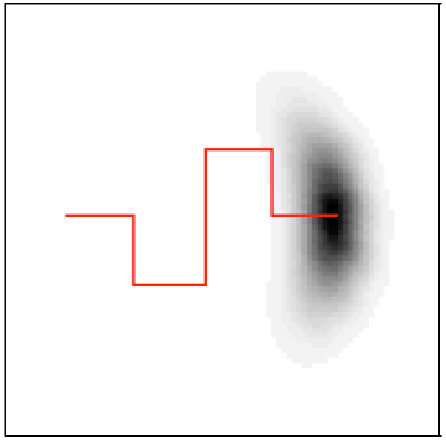
The Problem of Robot Localization

Given a map of the environment, how can a robot determine its pose (planar coordinates + orientation)?

Two sources of uncertainty:

- observations depend probabilistically on robot pose
- pose changes depend probabilistically on robot actions





<u>Example</u>:
Uncertainty of robot position after travelling along red path (shaded area indicates probability distribution)

Slides on Robot Localization are partly adapted from

Sebastian Thrun, http://www-2.cs.cmu.edu/~thrun/papers/thrun.probrob.html
Michael Beetz, http://wwwradig.in.tum.de/vorlesungen/as.SS03/folien_links.html

Formalization of Localization Problem

m	**model of environment (e.g. map)**
s_t	**pose at time t**
o_t	**observation at time t**
a_t	**action at time t**
$d_{0...t}$	**$= o_0, a_0, o_1, a_1, \ldots , o_t, a_t$** **observation and action data up to t**

<u>Task</u>: Estimate $p(s_t \mid d_{0...t}, m) = b_t(s_t)$ "robot´s belief state at time t"

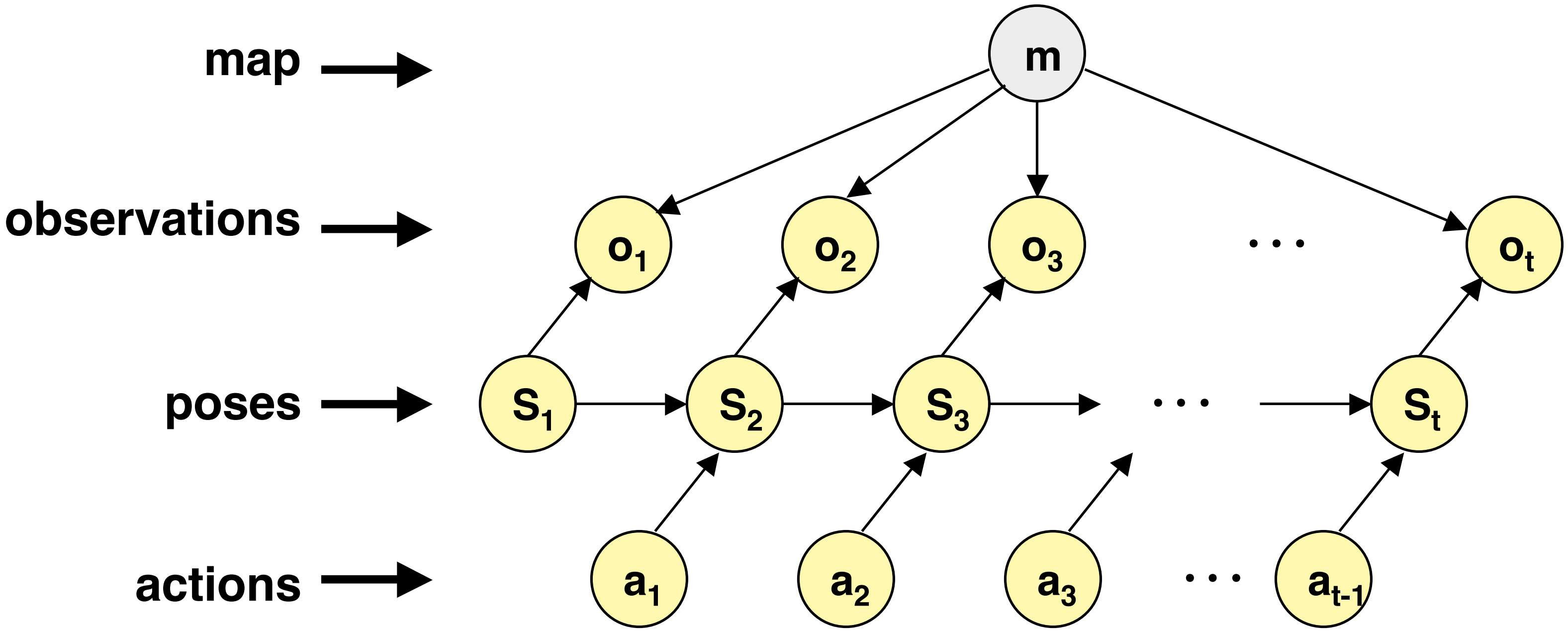
<u>Markov properties</u>:

- **Current observation depends only on current pose**
- **Next pose depends only on current pose and current action**

"Future is independent of past given current state"

Markov assumption implies static environment!
(Violation, for example, by robot actions changing the environment)

Structure of Probabilistic Localization



Recursive Markov Localization

$$b_t(s_t) = p(s_t \mid o_0, a_0, o_1, a_1, \ldots, a_{t-1}, o_t, m)$$

$$\overset{\text{Bayes}}{=} \alpha_t \, p(o_t \mid o_0, a_0, \ldots, a_{t-1}, s_t, m) \; p(s_t \mid o_0, a_0, \ldots, a_{t-1}, m)$$

$$\overset{\text{Markov}}{=} \alpha_t \, p(o_t \mid s_t, m) \; p(s_t \mid o_0, a_0, \ldots, a_{t-1}, m)$$

$$\overset{\text{Total Prob.}}{=} \alpha_t \, p(o_t \mid s_t, m) \int p(s_t \mid o_0, a_0, \ldots, a_{t-1}, s_{t-1}, m) \; p(s_{t-1} \mid o_0, a_0, \ldots, a_{t-1}, m) \, ds_{t-1}$$

$$\overset{\text{Markov}}{=} \alpha_t \, p(o_t \mid s_t, m) \int p(s_t \mid a_{t-1}, s_{t-1}, m) \; p(s_{t-1} \mid o_0, a_0, \ldots, a_{t-1}, m) \, ds_{t-1}$$

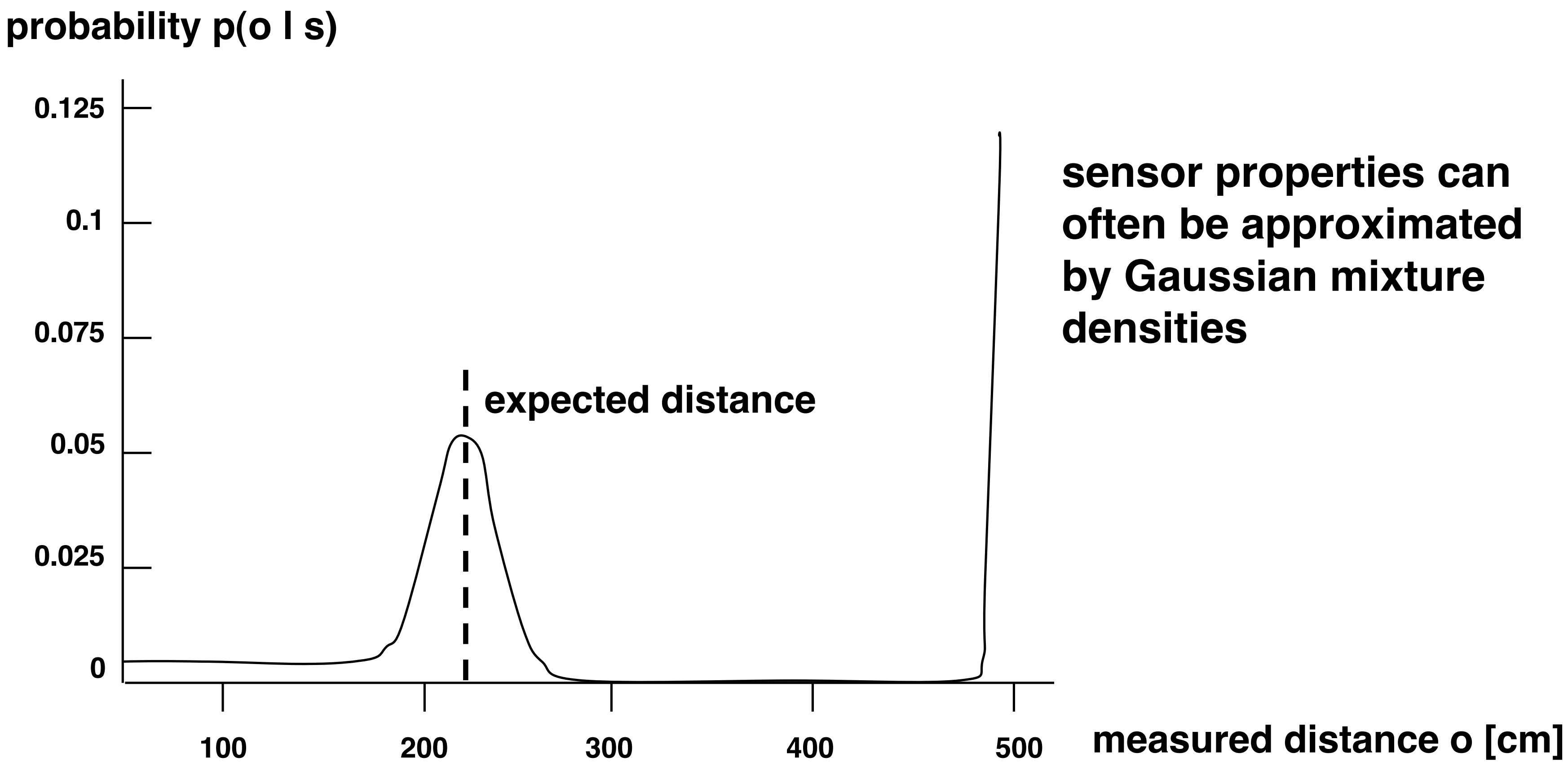
$$= \alpha_t \, p(o_t \mid s_t, m) \int p(s_t \mid a_{t-1}, s_{t-1}, m) \; b_{t-1}(s_{t-1}) \, ds_{t-1}$$

α_t is normalizing factor

$$b_t(s_t) = \alpha_t \, p(o_t \mid s_t, m) \int p(s_t \mid a_{t-1}, s_{t-1}, m) \; b_{t-1}(s_{t-1}) \, ds_{t-1}$$

$p(o_t \mid s_t, m)$	probabilistic perceptual model - often time-invariant: $p(o \mid s, m)$	must be specified for a specific robot and a specific environment
$p(s_t \mid a_{t-1}, s_{t-1}, m)$	probabilistic motion model - often time-invariant: $p(s' \mid a, s, m)$	

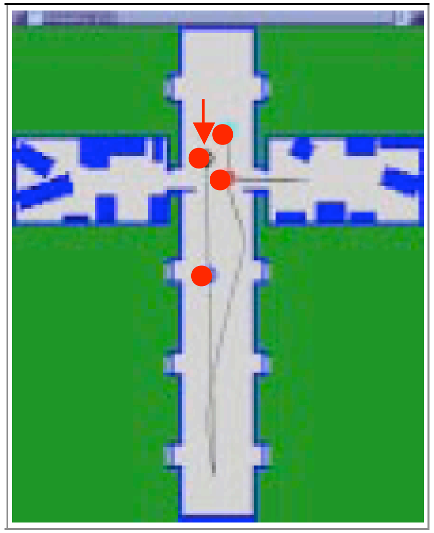
Probabilistic Sensor Model for Laser Range Finder



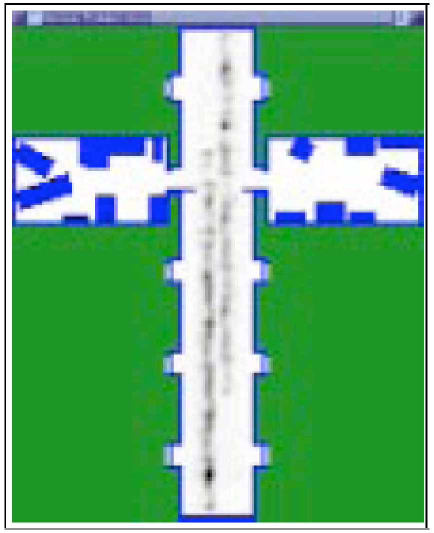
sensor properties can often be approximated by Gaussian mixture densities

Adapted from: Sebastian Thrun, Probabilistic Algorithms in Robotics
http://www-2.cs.cmu.edu/~thrun/papers/thrun.probrob.html

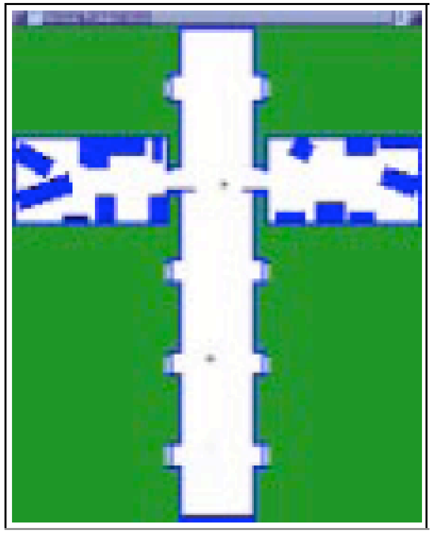
Grid-based Markov Localization (Example 1)



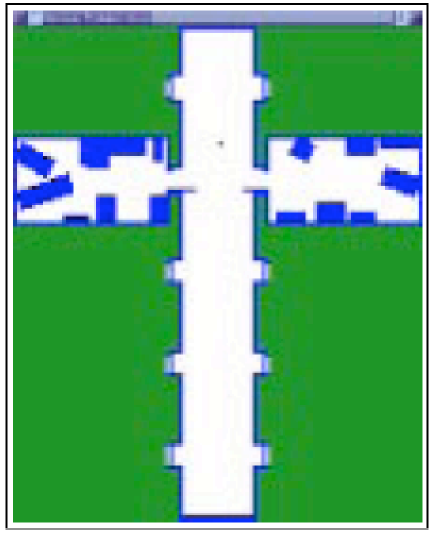
robot path with 4 reference poses, initially belief is equally distributed



distribution of belief at second pose



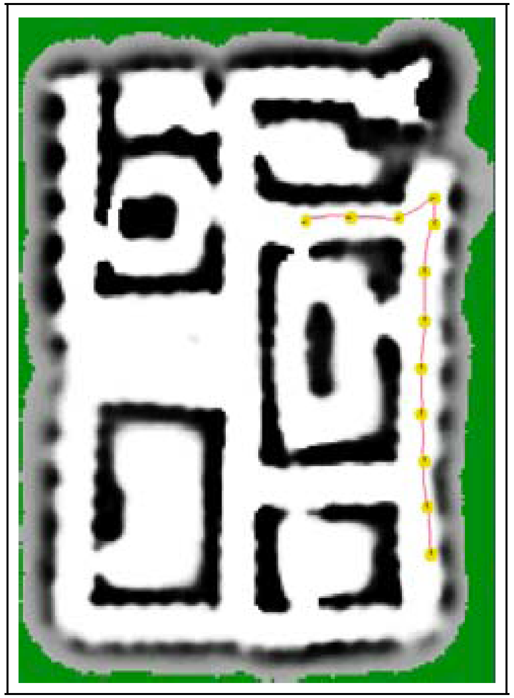
distribution of belief at third pose



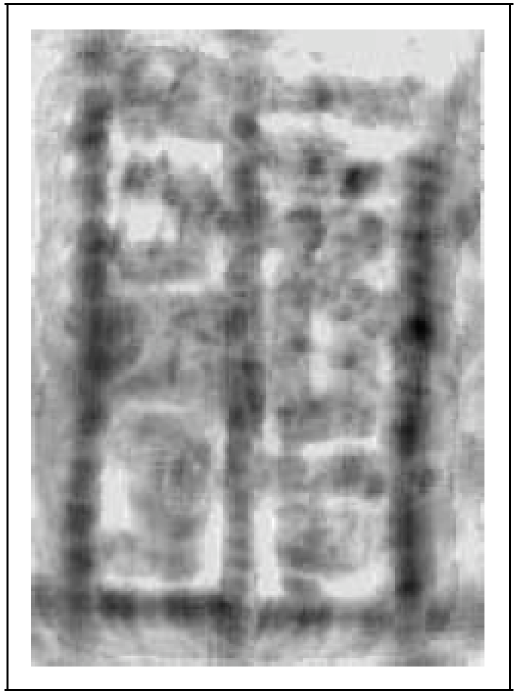
distribution of belief at fourth pose

Ambiguous localizations due to a repetitive and symmetric environment are sharpened and disambiguated after several observations.

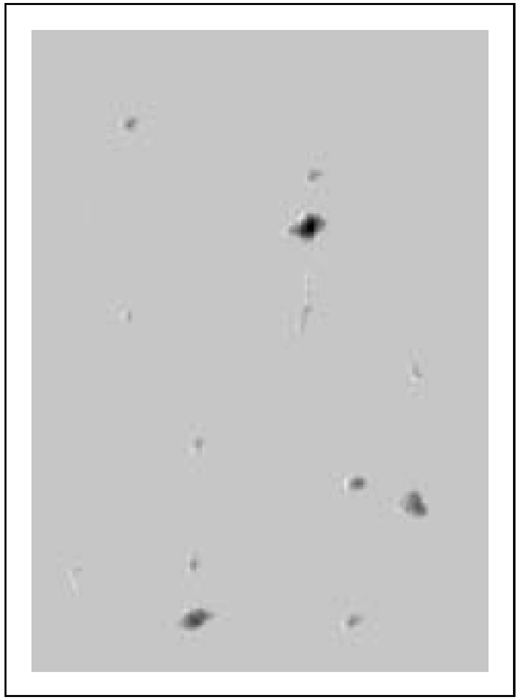
Grid-based Markov Localization (Example 2)



map and robot path



maximum position probabilities after 6 steps



maximum position probabilities after 12 steps

[Burgard et al. 96]

Approximating Probabilistic Update by Monte Carlo Localization (MCL)

"Importance Sampling" "Particle Filters" "Condensation Algorithm"	}	different names for a method to approximate a probability density by discrete samples (see slide "Sampling Methods")

Approximate implementation of belief update equation

$$b_t(s_t) = \alpha_t \, p(o_t \mid s_t, m) \int p(s_t \mid a_{t-1}, s_{t-1}, m) \; b_{t-1}(s_{t-1}) \, ds_{t-1}$$

1. Draw a sample s_{t-1} from the current belief $b_{t-1}(s_{t-1})$ with a likelihood given by the importance factors of the belief $b_{t-1}(s_{t-1})$.
2. For this s_{t-1} guess a successor pose s_t according to the distribution $p(s_t \mid a_{t-1}, s_{t-1}, m)$.
3. Assign a preliminary importance factor $p(o_t \mid s_t, m)$ to this sample and assign it to the new sample representing $b_t(s_t)$.
4. Repeat Step 1 through 3 m times. Finally, normalize the importance factors in the new sample set $b_t(s_t)$ so that they add up to 1.

MCL is very effective and can give good results with as few as 100 samples.

Simultaneous Localization and Mapping (SLAM)

Typical problem for a mobile robot in an unknown environment:

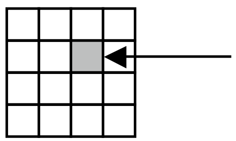
- **learn the environment ("mapping")**
- **keep track of position and orientation ("localization")**

"Chicken-and-egg" problem:

- **robot needs knowledge of environment in order to interpret sensor readings for localization**
- **robot needs pose knowledge in order to interpret sensor readings for mapping**

Make the environment a multidimensional probabilistic variable!

Example: Model of environment is a probabilistic occupancy grid



$$P_{ij} = \begin{cases} e_{ij} & (x_i, y_i) \text{ is empty} \\ 1-e_{ij} & (x_i, y_i) \text{ is occupied} \end{cases}$$

Bayes Filter for SLAM

Extend the localization approach to simultaneous mapping:

$$b_t(s_t) = \alpha_t\, p(o_t \mid s_t, m) \int p(s_t \mid a_{t-1}, s_{t-1}, m)\; b_{t-1}(s_{t-1})\, ds_{t-1}$$

$$b_t(s_t, m_t) = \alpha_t\, p(o_t \mid s_t, m_t) \iint p(s_t, m_t \mid a_{t-1}, s_{t-1}, m_{t-1})\; b_{t-1}(s_{t-1}, m_{t-1})\; ds_{t-1}\, dm_{t-1}$$

Assuming a time-invariant map and map-independent motion:

$$b_t(s_t, m) = \alpha_t\, p(o_t \mid s_t, m) \int p(s_t \mid a_{t-1}, s_{t-1})\; b_{t-1}(s_{t-1}, m)\, ds_{t-1}$$

$b_t(s_t, m)$ is (N+3)-dimensional with N variables for m (N >> 1000) and 3 for s_t

=> complexity problem

Important approaches to cope with this complexity:

- **Kalman filtering (Gaussian probabilities and linear updating)**
- **estimating only the mode of the posterior, $\text{argmax}_m\, b(m)$**
- **treating the robot path as "missing variables" in Expectation Maximization**

Kalman Filter for SLAM Problems (1)

Basic Kalman Filter assumptions:

1. Next-state function is linear with added Gaussian noise
2. Perceptual model is linear with added Gaussian noise
3. Initial uncertainty is Gaussian

Ad 1) Next state in SLAM is pose s_t and model m.

- m is assumed constant
- s_t is non-linear in general, approximately linear in a first-degree Taylor series expansion ("Extended Kalman Filtering")

Let x_t be the state variable (s_t, m) and $\varepsilon_{control}$ Gaussian noise with covariance $\Sigma_{control}$, then

$$p(x_t \mid a_{t-1}, x_{t-1}) = A\, x_{t-1} + B\, a_{t-1} + \varepsilon_{control}$$

Ad 2) Sensor measurements are usually nonlinear in robotics, with non-white Gaussian noise. Approximation by first-degree Taylor series and $\varepsilon_{measure}$ Gaussian noise with covariance $\Sigma_{measure}$.

$$p(o_t \mid x_t) = C\, x_t + \varepsilon_{measure}$$

Kalman Filter for SLAM Problems (2)

Bayes Filter equation

$$b_t(s_t, m) = \alpha_t\, p(o_t \mid s_t, m) \int p(s_t, m_t \mid a_{t-1}, s_{t-1}, m)\; b_{t-1}(s_{t-1}, m)\, ds_{t-1}$$

can be rewritten using the standard Kalman Filter equations:

$$\mu_{t-1} = \mu_{t-1} + B\, a_t$$

$$\Sigma'_{t-1} = \Sigma_{t-1} + \Sigma_{control}$$

$$K_t = \Sigma'_{t-1}\, C^T \left(C \Sigma'_{t-1} C^T + \Sigma_{measure}\right)^{-1}$$

$$\mu_t = \mu'_{t-1} + K_t \left(o_{t-1} - C \mu'_{t-1}\right)$$

$$\Sigma_t = (I - K_t\, C)\, \Sigma'_{t-1}$$

Compare with slides on Kalman Filtering in "Bildverarbeitung".

- **Kalman Filtering estimates the full posterior distribution for all poses (not only the maximum)**
- **Guaranteed convergence to true map and robot pose**
- **Gaussian sensor noise is a bad model for correspondence problems**

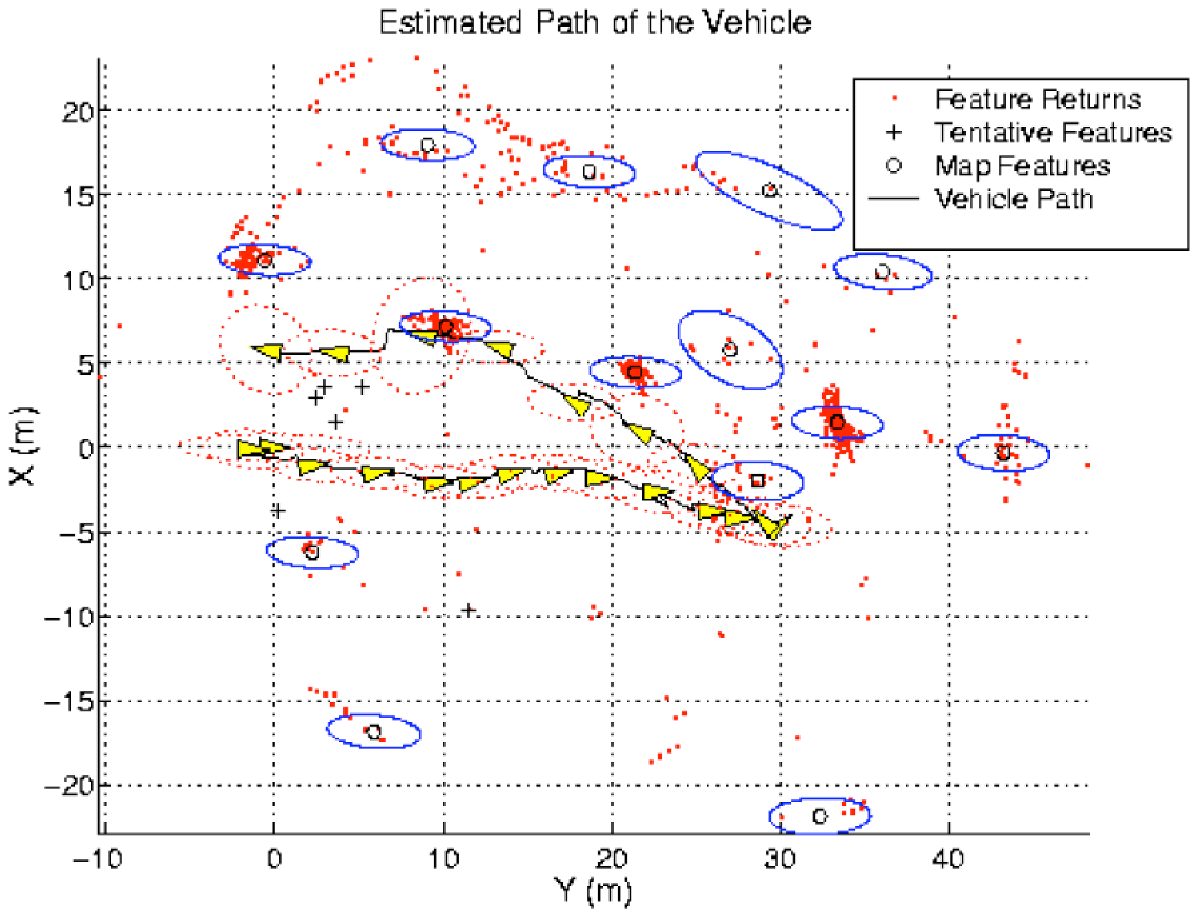
Example: Underwater Sonic Mapping

From: S.Williams, G. Dissanayake, and H.F. Durrant-Whyte. Towards terrain-aided navigation for underwater robotics. *Advanced Robotics*, 15(5), 2001.

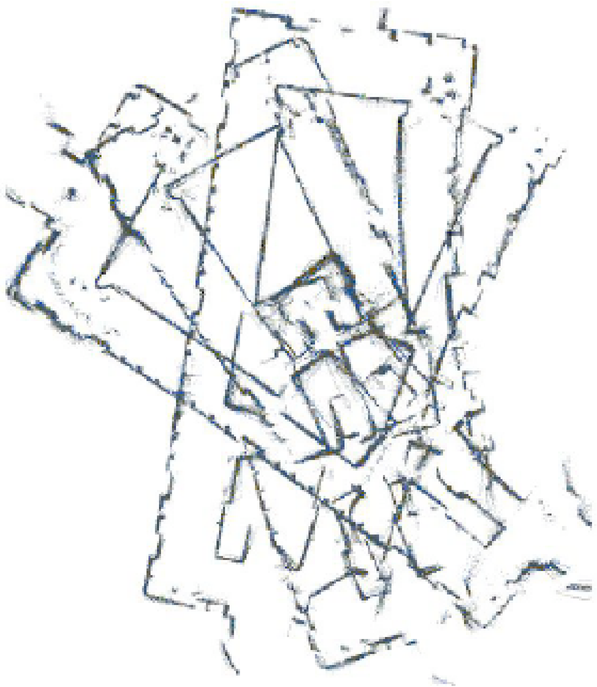
Kalman Filter map and pose estimation

Figure shows:

- **estimated path of underwater vehicle with ellipses indicating position uncertainty**
- **14 landmarks obtained by sonar measurements with ellipses indicating uncertainty, 5 artificial landmarks, the rest other reflective objects**
- **additional dots for weak landmark hypotheses**

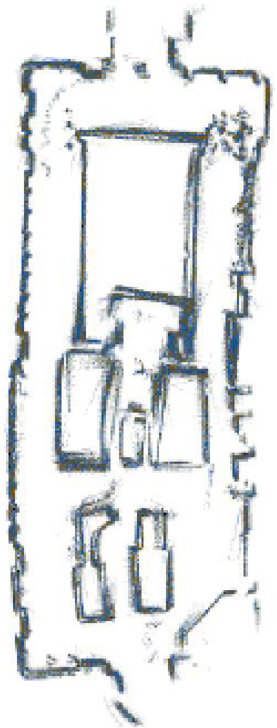


Solving the Correspondence Problem



Map obtained from raw sensory data of a cyclic environment (large hall of a museum) based on robot´s odometry

correspondence problem!



Map obtained by EM algorithm: Iterative maximization of both robot path and model

non-incremental procedure!

Mapping with Expectation Maximization

Principle of EM mapping algorithm:

> **Repeat until no more changes**
>
> **E-step: Estimate robot poses for given map**
>
> **M-step: Calculate most likely map given poses**

The algorithm computes the maximum of the expectation of the joint log likelihood of the data $d^t = \{a_0, o_0, \ldots, a_t, o_t\}$ and the robot´s path $s^t = \{s_0, \ldots, s_t\}$.

$$m^{[i+1]} = \arg\max_{m} E_{s^t}\left[\log p\left(d^t, s^t \mid m\right) \mid m^{[i]}, d^t\right]$$

$$m^{[i+1]} = \arg\max_{m} \sum_{\tau} \int p\left(s_\tau \mid m^{[i]}, d^t\right) \log p(o_\tau \mid s_\tau, m)\, ds_\tau$$

E-step: Compute the posterior of pose s_τ based on $m^{[i]}$ and <u>all</u> data including $t > \tau$: => different from incremental localization

M-step: Maximize $\log p(o_\tau \mid s_\tau, m)$ for all τ and all poses s^t under the expectation calculated in the E-step

Mapping with Incremental Maximum-Likelihood Estimation

Stepwise maximum-likelihood estimation of map and pose is inferior to Kalman Filtering and EM estimation, but less complex.

Obtain series of maximum-likelihood maps and poses

$$m_1^*, m_2^*, \ldots$$

$$s_1^*, s_2^*, \ldots$$

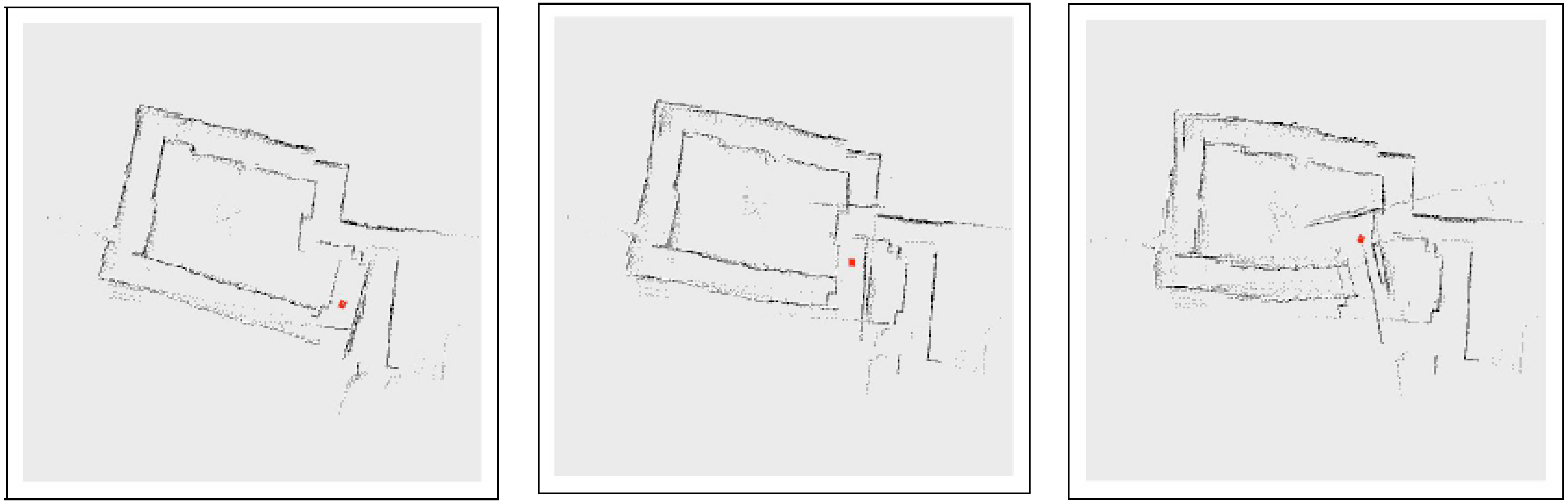
by maximizing the marginal likelihood:

$$\langle m_t^*, s_t^* \rangle = \underset{m_t,\, s_t}{\operatorname{argmax}}\; p(o_t \mid m_t, s_t)\; p(m_t, s_t \mid a_t, s_{t-1}^*, m_{t-1}^*)$$

This equation follows from the Bayes Filter equation by assuming that map and pose at t-1 are known for certain.

- **real-time computation possible**
- **unable to handle cycles**

Example of Incremental Maximum-Likelihood Mapping



At every time step, the map is grown by finding the most likely continuation. Map estimates do not converge as robot completes cycle because of accumulated pose estimation error.

Examples from: Sebastian Thrun, Probabilistic Algorithms in Robotics
http://www-2.cs.cmu.edu/~thrun/papers/thrun.probrob.html

Maintaining a Pose Posterior Distribution

Problems with cyclic environment can be overcome by maintaining not only the maximum-likelihood pose estimate at t-1 but also the uncertainty distribution using Bayes Filter:

$$p(s_t \mid o^t, a^t) = \alpha \, p(o_t \mid s_t) \int p(s_t \mid a_{t-1}, s_{t-1}) \; p(s_{t-1} \mid o^{t-1}, a^{t-1}) \, ds_{t-1}$$

Last example repeated, representing the pose posterior by particles. Uncertainty is transferred onto map, so major corrections remain possible.