Robotics

Chapter 25

Outline

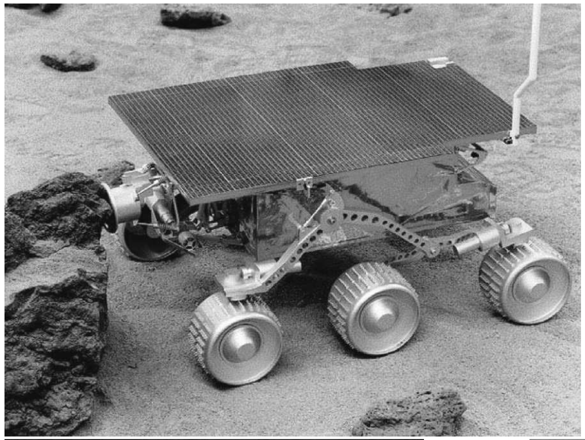
Robots, Effectors, and Sensors

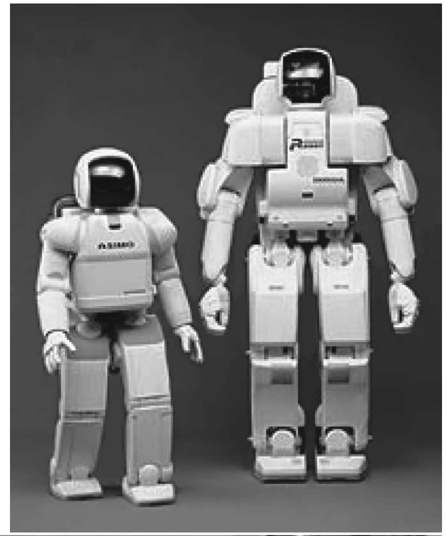
Localization and Mapping

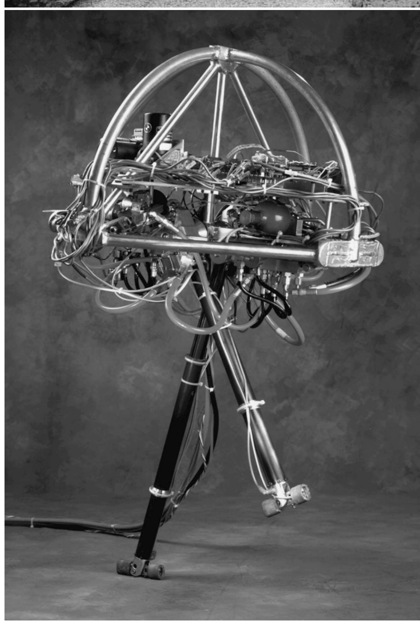
Motion Planning

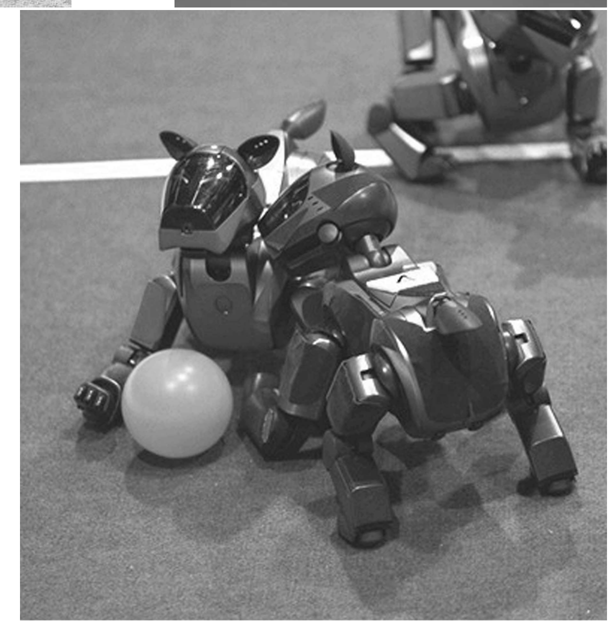
Motor Control

Mobile Robots

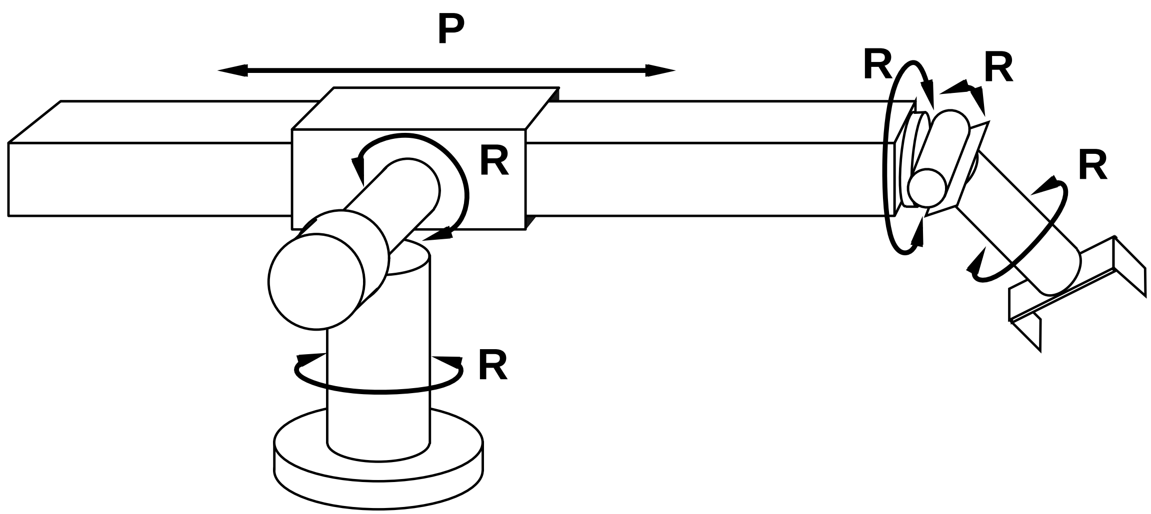








Manipulators

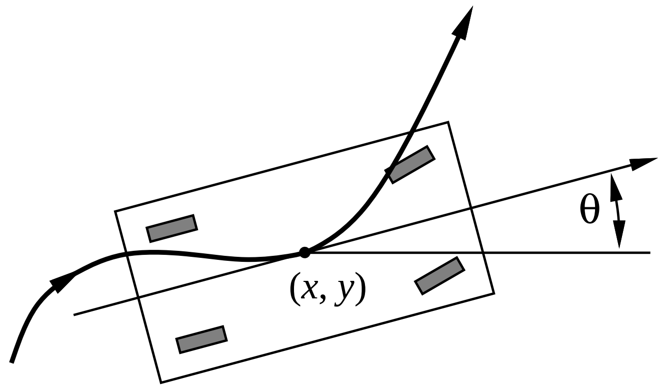


Configuration of robot specified by 6 numbers

$\Rightarrow$ 6 degrees of freedom (DOF)

6 is the minimum number required to position end-effector arbitrarily.
For dynamical systems, add velocity for each DOF.

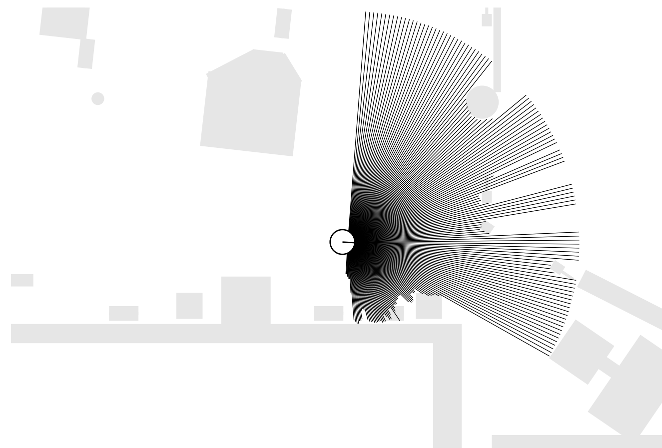
Non-holonomic robots



A car has more DOF (3) than controls (2), so is non-holonomic;
cannot generally transition between two infinitesimally close configurations

Sensors

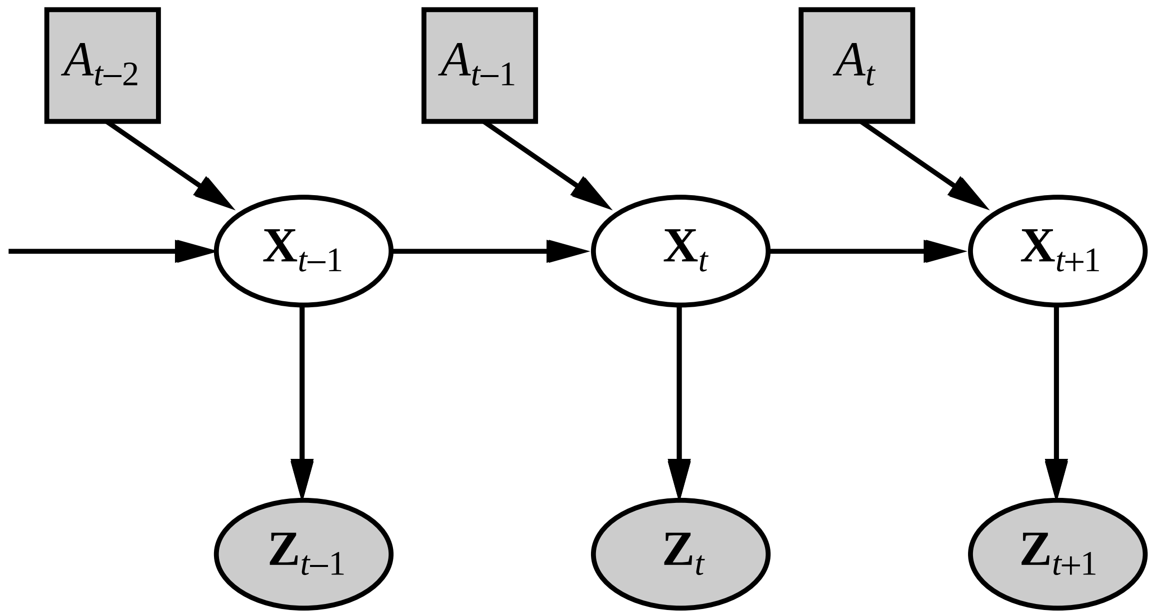
Range finders: sonar (land, underwater), laser range finder, radar (aircraft), tactile sensors, GPS



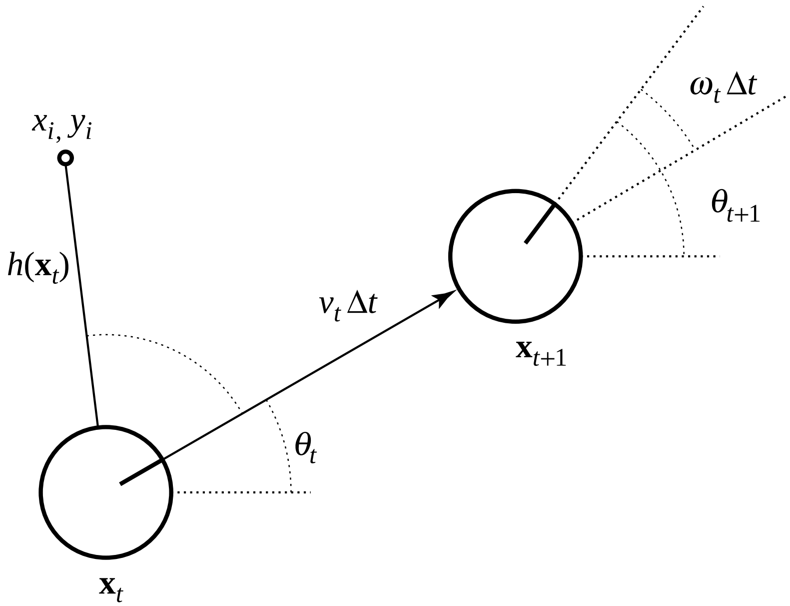
Imaging sensors: cameras (visual, infrared)
Proprioceptive sensors: shaft decoders (joints, wheels), inertial sensors, force sensors, torque sensors

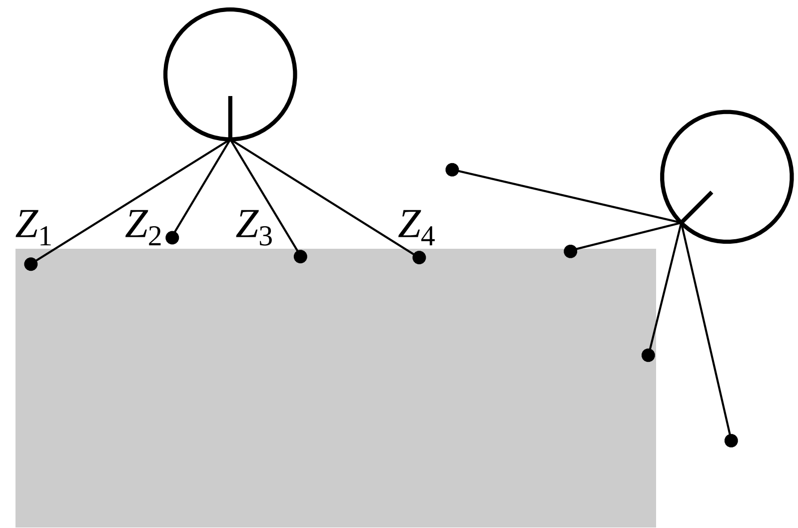
Localization—Where Am I?

Compute current location and orientation (pose) given observations:



Localization contd.

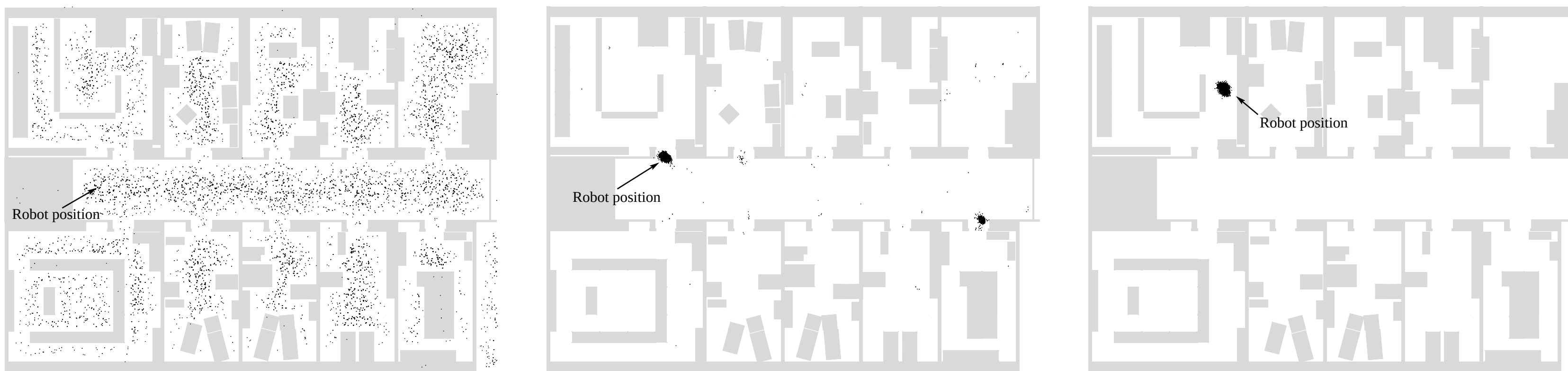




Assume Gaussian noise in motion prediction, sensor range measurements

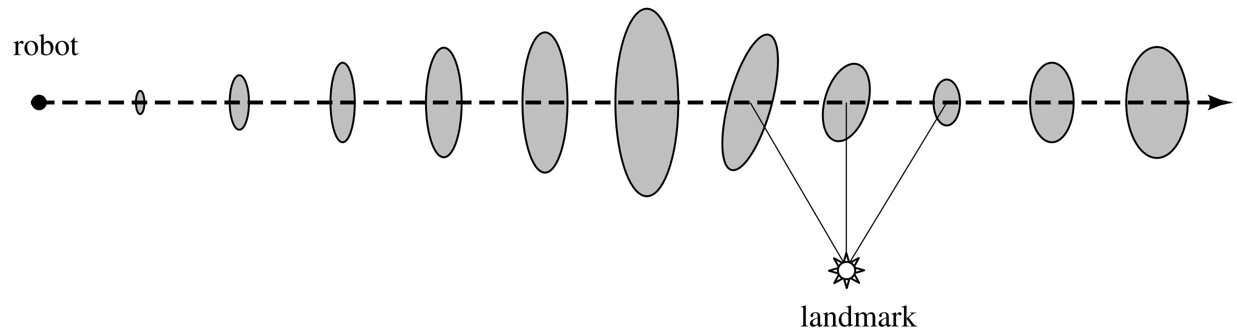
Localization contd.

Can use particle filtering to produce approximate position estimate



Localization contd.

Can also use extended Kalman filter for simple cases:



Assumes that landmarks are *identifiable*—otherwise, posterior is multimodal

Mapping

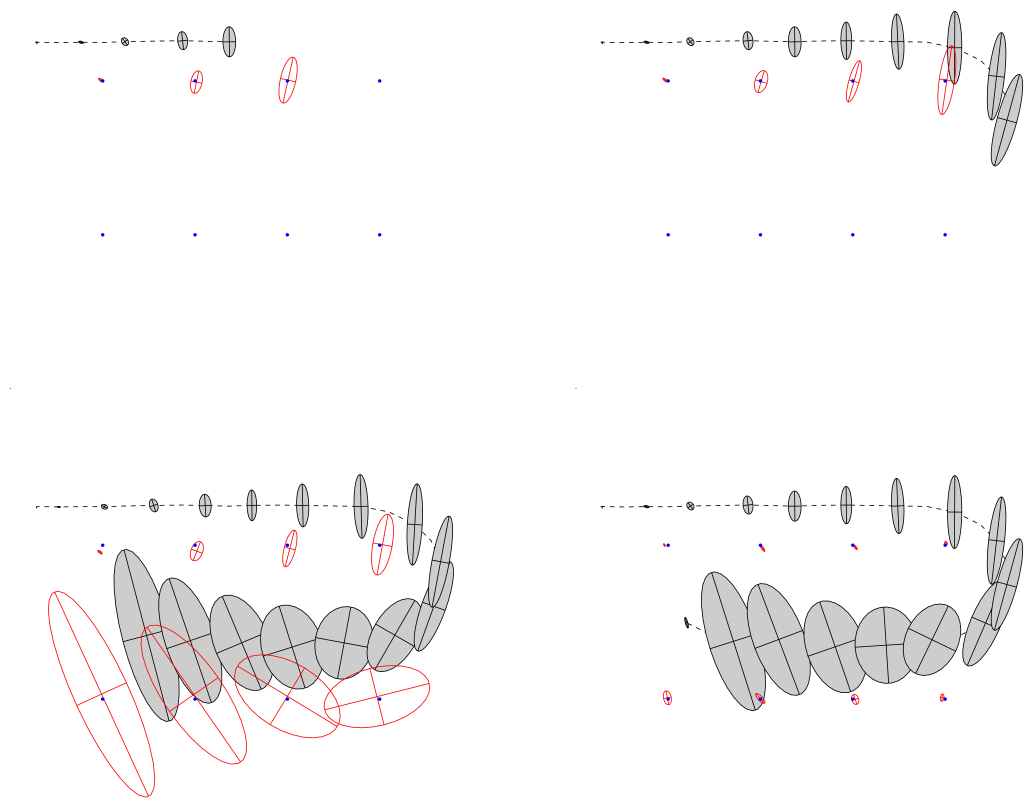
Localization: given map and observed landmarks, update pose distribution

Mapping: given pose and observed landmarks, update map distribution

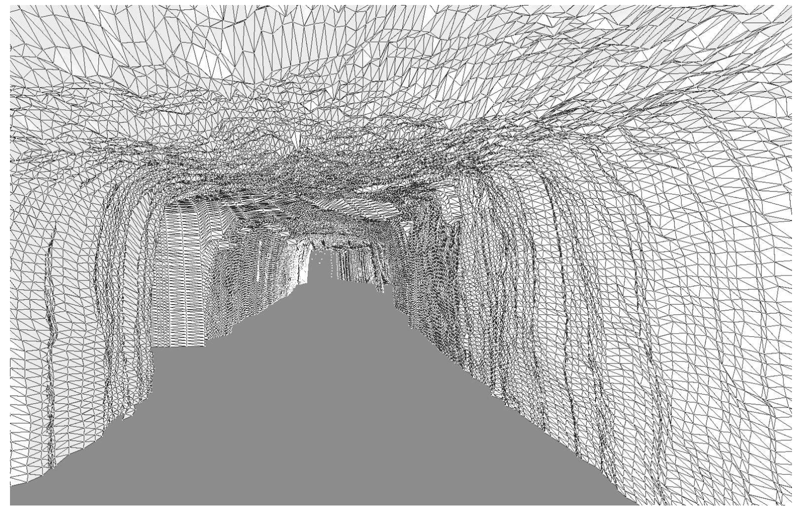
SLAM: given observed landmarks, update pose and map distribution

Probabilistic formulation of SLAM:
add landmark locations $L_1, \ldots, L_k$ to the state vector,
proceed as for localization

Mapping contd.

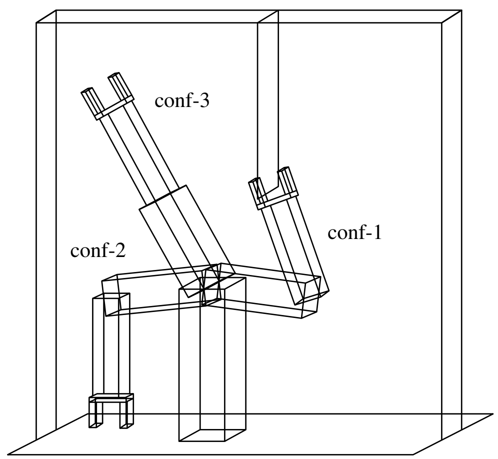


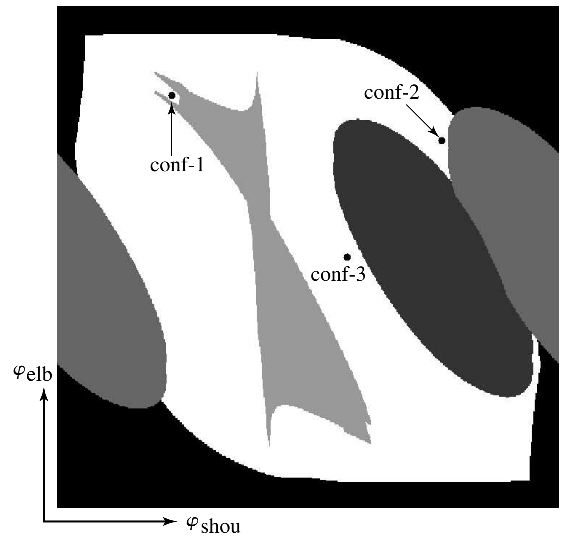
3D Mapping example



Motion Planning

Idea: plan in configuration space defined by the robot's DOFs





Solution is a point trajectory in free C-space

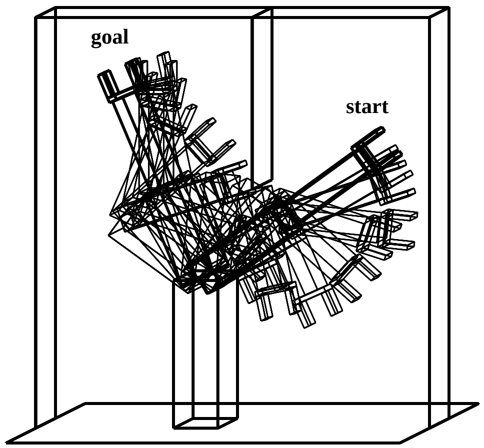
Configuration space planning

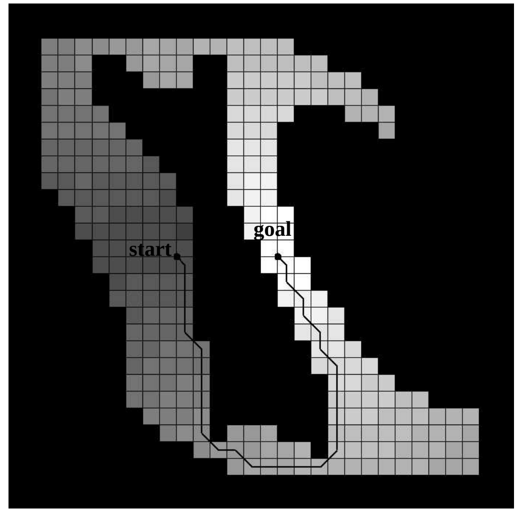
Basic problem: ∞^d states! Convert to **finite** state space.

Cell decomposition:
divide up space into simple cells,
each of which can be traversed "easily" (e.g., convex)

Skeletonization:
identify finite number of easily connected points/lines
that form a graph such that any two points are connected
by a path on the graph

Cell decomposition example



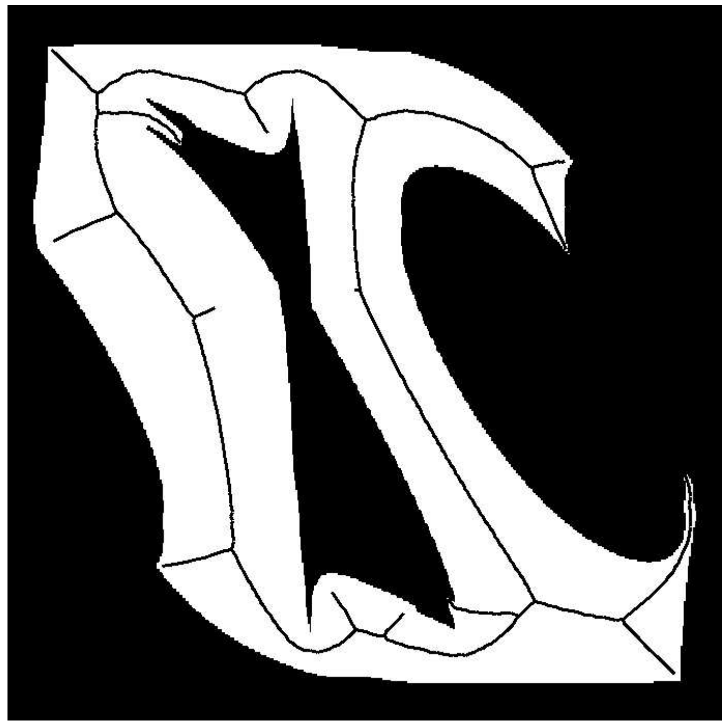


Problem: may be no path in pure freespace cells

Solution: recursive decomposition of mixed (free+obstacle) cells

Skeletonization: Voronoi diagram

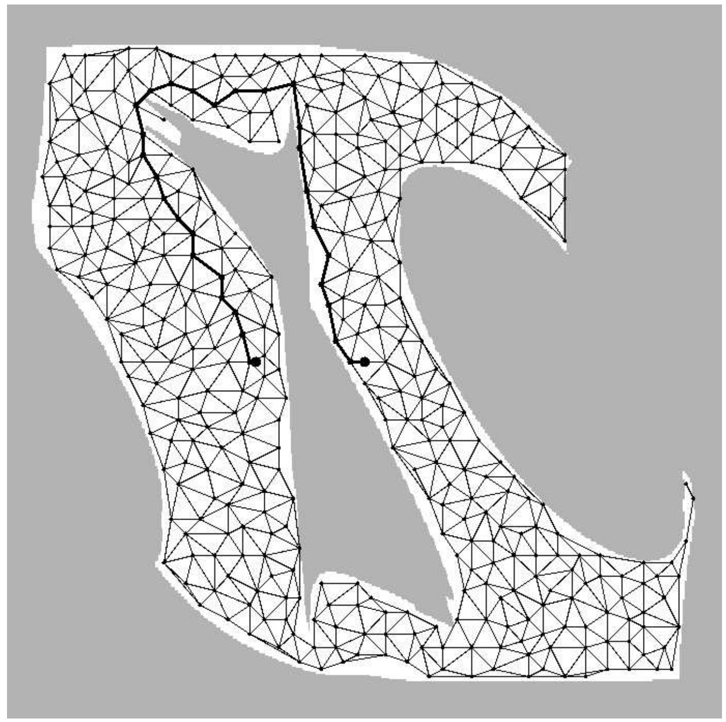
Voronoi diagram: locus of points equidistant from obstacles



Problem: doesn't scale well to higher dimensions

Skeletonization: Probabilistic Roadmap

A probabilistic roadmap is generated by generating random points in C-space and keeping those in freespace; create graph by joining pairs by straight lines



Problem: need to generate enough points to ensure that every start/goal pair is connected through the graph

Motor control

Can view the motor control problem as a search problem
in the dynamic rather than kinematic state space:

– state space defined by $x_1, x_2, \ldots, \dot{x}_1, \dot{x}_2, \ldots$

– continuous, high-dimensional (Sarcos humanoid: 162 dimensions)

Deterministic control: many problems are exactly solvable
esp. if linear, low-dimensional, exactly known, observable

Simple regulatory control laws are effective for specified motions

Stochastic optimal control: very few problems exactly solvable

$\Rightarrow$ approximate/adaptive methods

Biological motor control

Motor control systems are characterized by massive redundancy

Infinitely many trajectories achieve any given task

E.g., 3-link arm moving in plane throwing at a target
    simple 12-parameter controller, one degree of freedom at target
    11-dimensional continuous space of optimal controllers

Idea: if the arm is noisy, only "one" optimal policy minimizes error at target

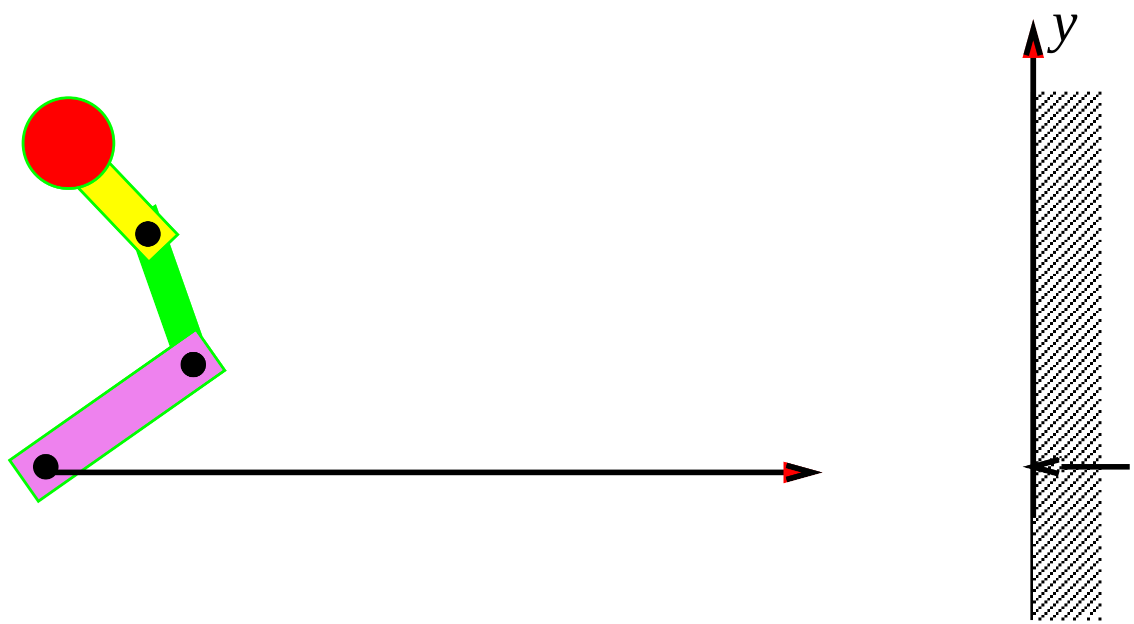
I.e., noise-tolerance might explain actual motor behaviour

Harris & Wolpert (*Nature*, 1998): signal-dependent noise
explains eye saccade velocity profile perfectly

Setup

Suppose a controller has "intended" control parameters θ_0 which are corrupted by noise, giving θ drawn from P_{θ_0}

Output (e.g., distance from target) $y = F(\theta)$;



Simple learning algorithm: Stochastic gradient

Minimize $E_\theta[y^2]$ by gradient descent:

$$\begin{aligned}
\nabla_{\theta_0} E_\theta[y^2] &= \nabla_{\theta_0} \int P_{\theta_0}(\theta) F(\theta)^2 d\theta \\
&= \int \frac{\nabla_{\theta_0} P_{\theta_0}(\theta)}{P_{\theta_0}(\theta)} F(\theta)^2 P_{\theta_0}(\theta) d\theta \\
&= E_\theta[\frac{\nabla_{\theta_0} P_{\theta_0}(\theta)}{P_{\theta_0}(\theta)} y^2]
\end{aligned}$$

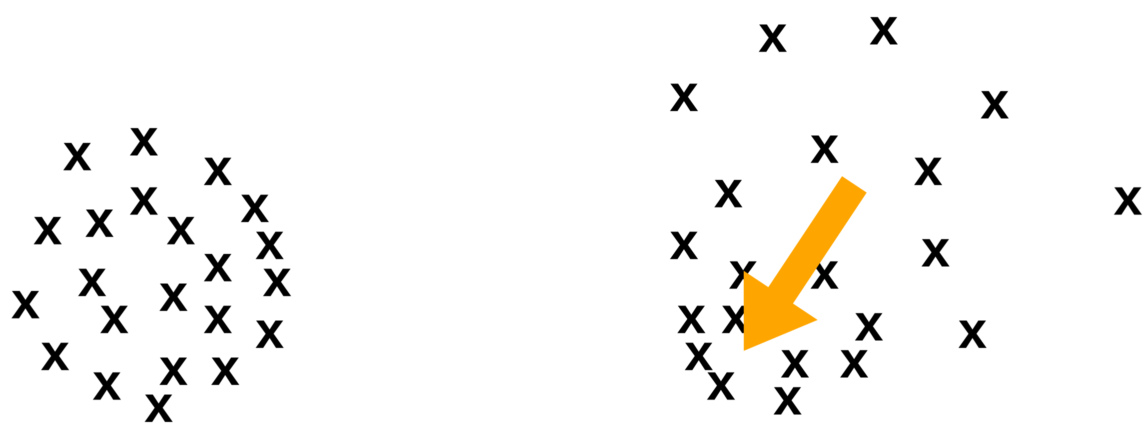
Given samples (θ_j, y_j), $j = 1, \ldots, N$, we have

$$\nabla_{\theta_0} \hat{E}_\theta[y^2] = \frac{1}{N} \sum_{j=1}^{N} \frac{\nabla_{\theta_0} P_{\theta_0}(\theta_j)}{P_{\theta_0}(\theta_j)} y_j^2$$

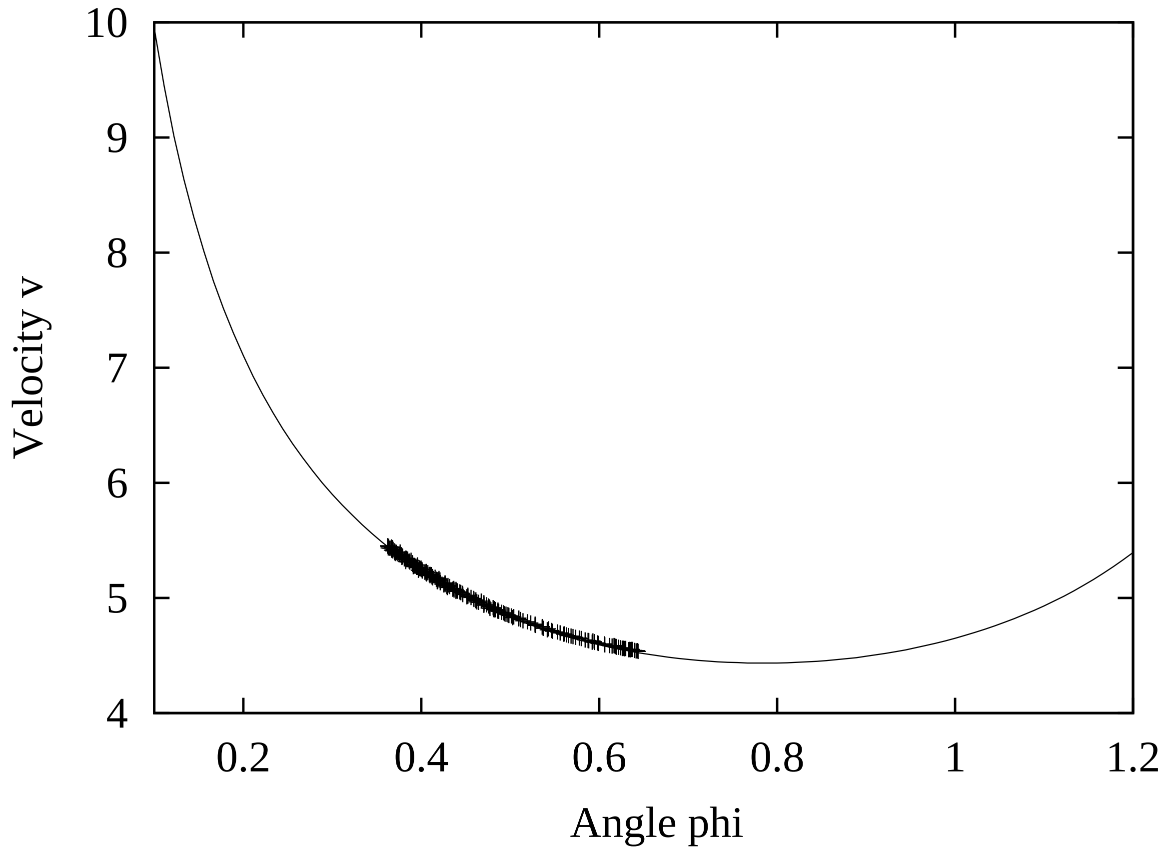
For Gaussian noise with covariance Σ, i.e., $P_{\theta_0}(\theta) = N(\theta_0, \Sigma)$, we obtain

$$\nabla_{\theta_0} \hat{E}_\theta[y^2] = \frac{1}{N} \sum_{j=1}^{N} \Sigma^{-1}(\theta_j - \theta_0) y_j^2$$

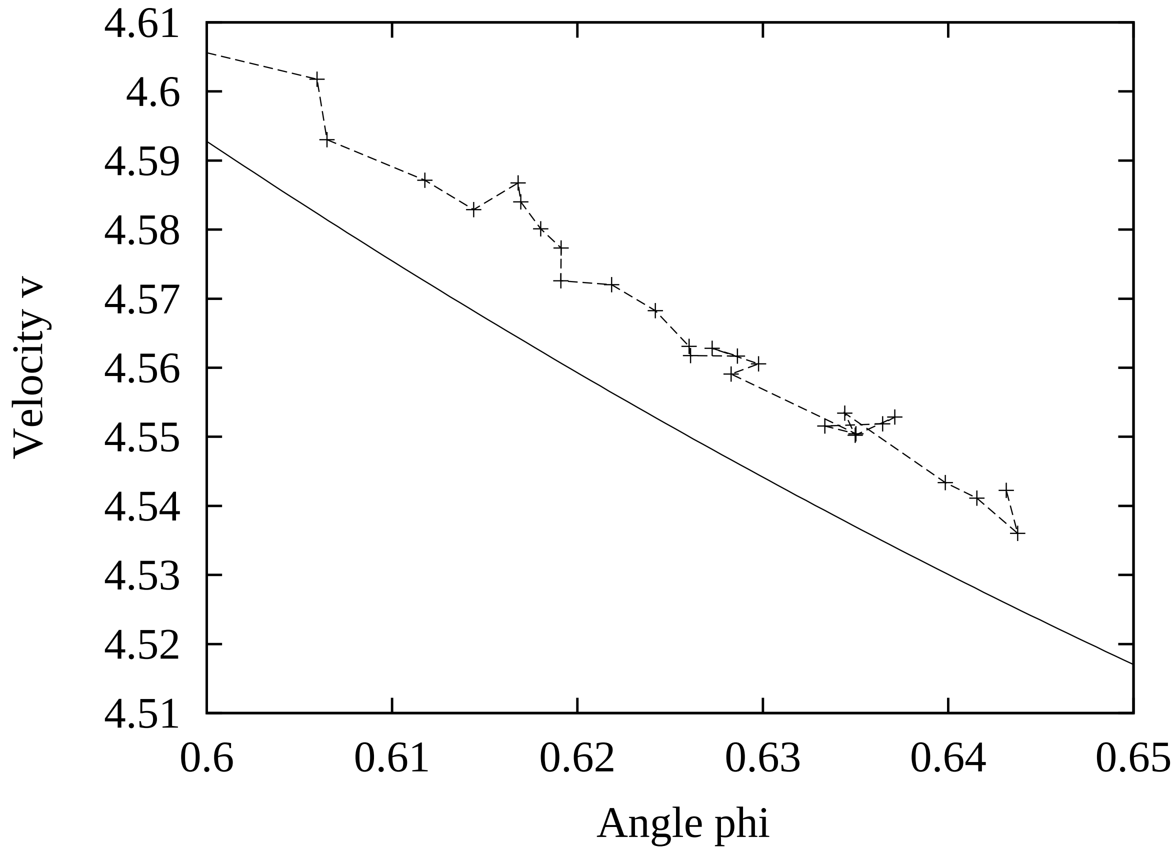
What the algorithm is doing



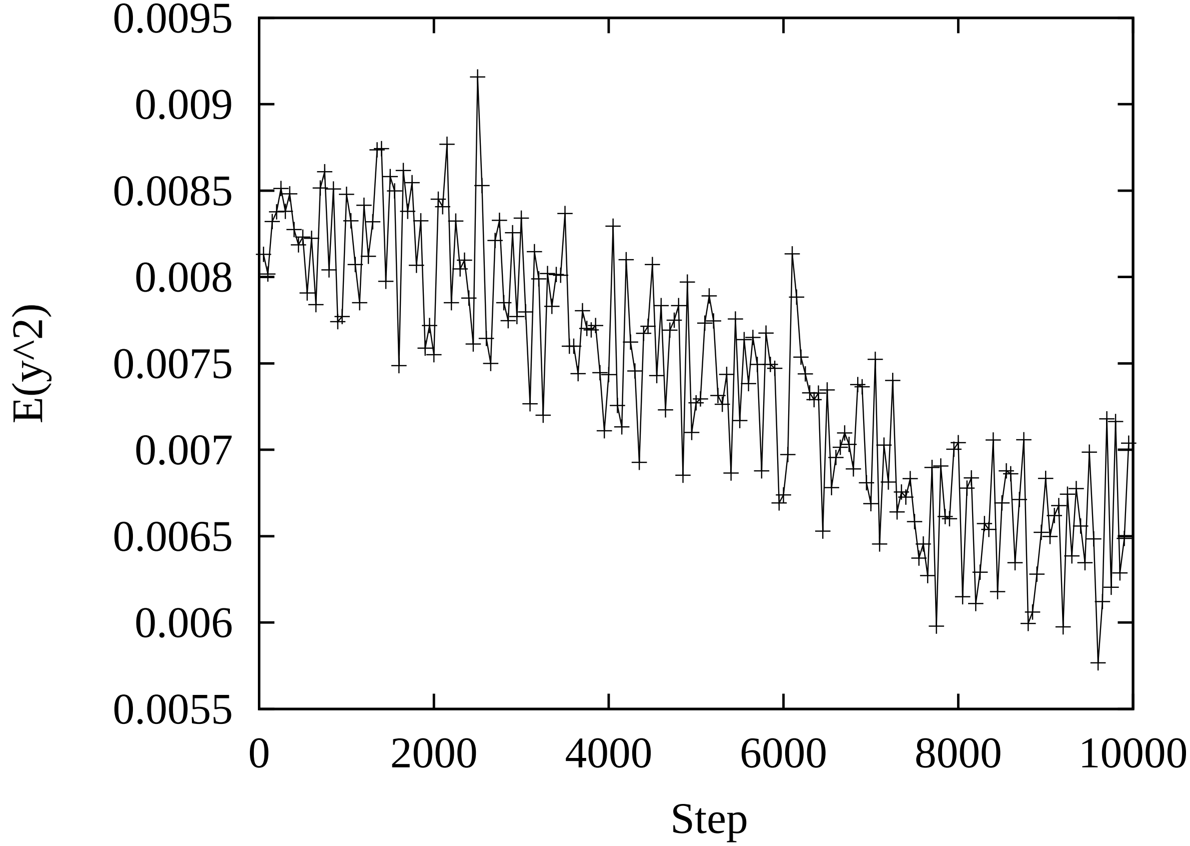
Results for 2–D controller



Results for 2–D controller



Results for 2–D controller



Summary

The rubber hits the road

Mobile robots and manipulators

Degrees of freedom to define robot configuration

Localization and mapping as probabilistic inference problems
(require good sensor and motion models)

Motion planning in configuration space
requires some method for finitization