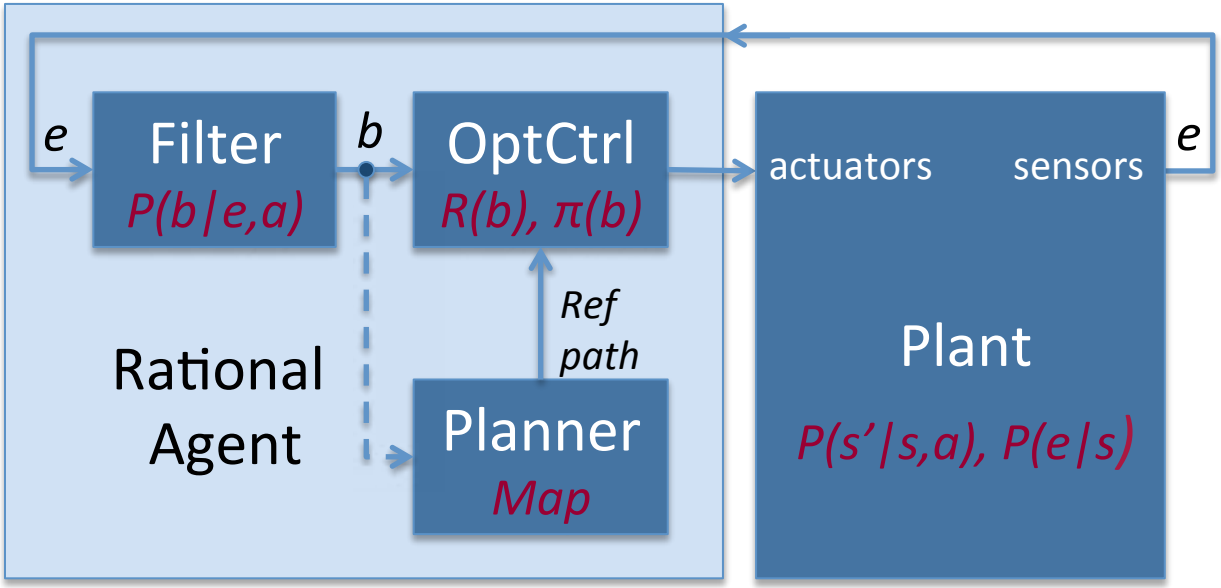
Robotics

Chapter 25



Outline

- Robots, Sensors, and Actuators
- Localization and Mapping
- Motion Planning
- Motor Control

Robots

Robots are agents manipulating their physical world

- Sensors: Allow them to perceive their environment
- Actuators: Assert physical forces on the environment

Three primary categories:

- Manipulators: Physically anchored to their workplace
- Mobile robots: Move around using wheels, legs, etc.
- Mobile manipulators: Combine mobility with manipulation

Manipulators (Robot Arms)

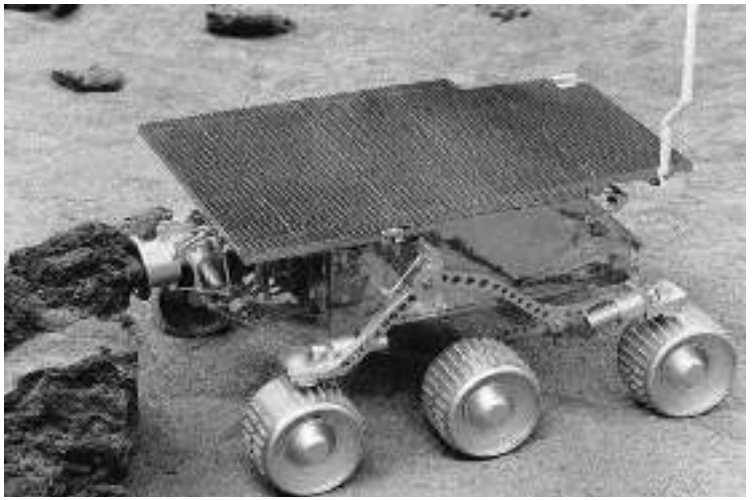
Industrial robotic manipulator
For stacking bags on a pallet

Surgical Vinci robots
In the operating room

Most common type of industrial robots: Around 1M word-wide

Mobile Robots

Unmanned ground vehicle (UGV)



Planetary rover

Unmanned air vehicle (UGV)

Autonomous underwater vehicle (AUV)

Mobile Manipulators

Humanoid P3 and Asimo (Honda)

Robocup dogs (Aibo)

Mobile Manipulator (W.Garage)

Sensors

Sensors are the perceptual interface to the environment

- Passive (camera): Capture signals generated by other sources
- Active (sonar): Send energy and analyze reflection (more info)

Range finders are sensors that measure distance

- Sonar: Emit directional sound waves which get reflected
- Optical: Emit active light signals and measure reflection time
- Tactile: Measure range based on physical contact

Sensors

Location sensors use range sensing to determine location

- GPS: Measures distance to satellites (31) that emit pulsed signals
- Triangulating signals from multiple satellites determine position
- Differential GPS: Use a 2nd ground receiver with known position

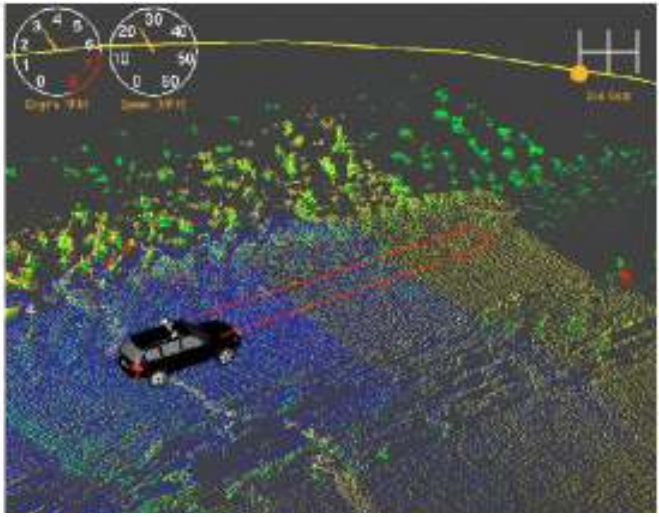
Proprioceptive sensors inform robot of its own motion

- Shaft decoders: Count revolution of motors in small increments
- Odometers: Measure the distance traveled (wheels tend to drift)
- Inertial sensors: Measure resistance of mass to acceleration
- Force and torque sensors: Used to handle fragile objects

Active Sensors

Time-of-flight camera and obtained image (Mesa)
Produces range images up to 60 per second

Laser scanner and obtained image (Sick)
Use laser beams and special 1-pixel cameras



Stanley: Five laser sensors at different angles downward. Each acquires a 3D point cloud (Thrun et al)

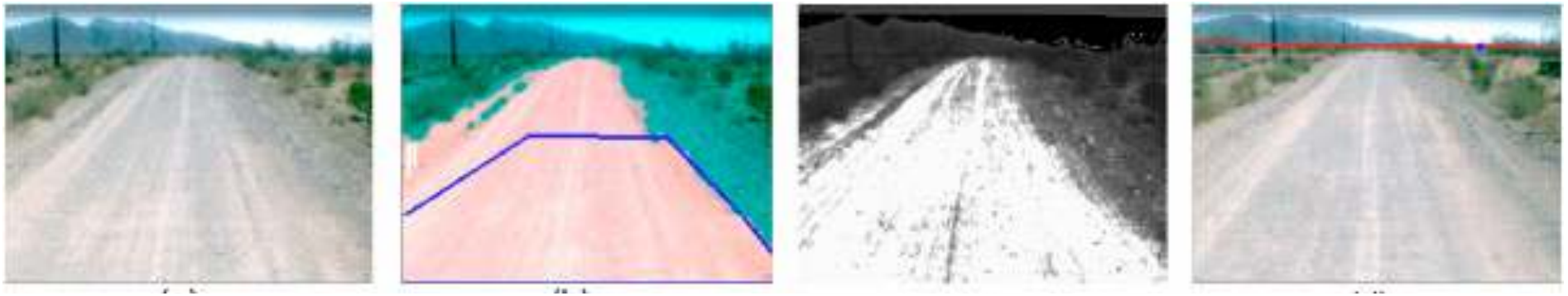
Passive Sensors

VC-50i
(Canon)

IMU (ARM UM6-LT)

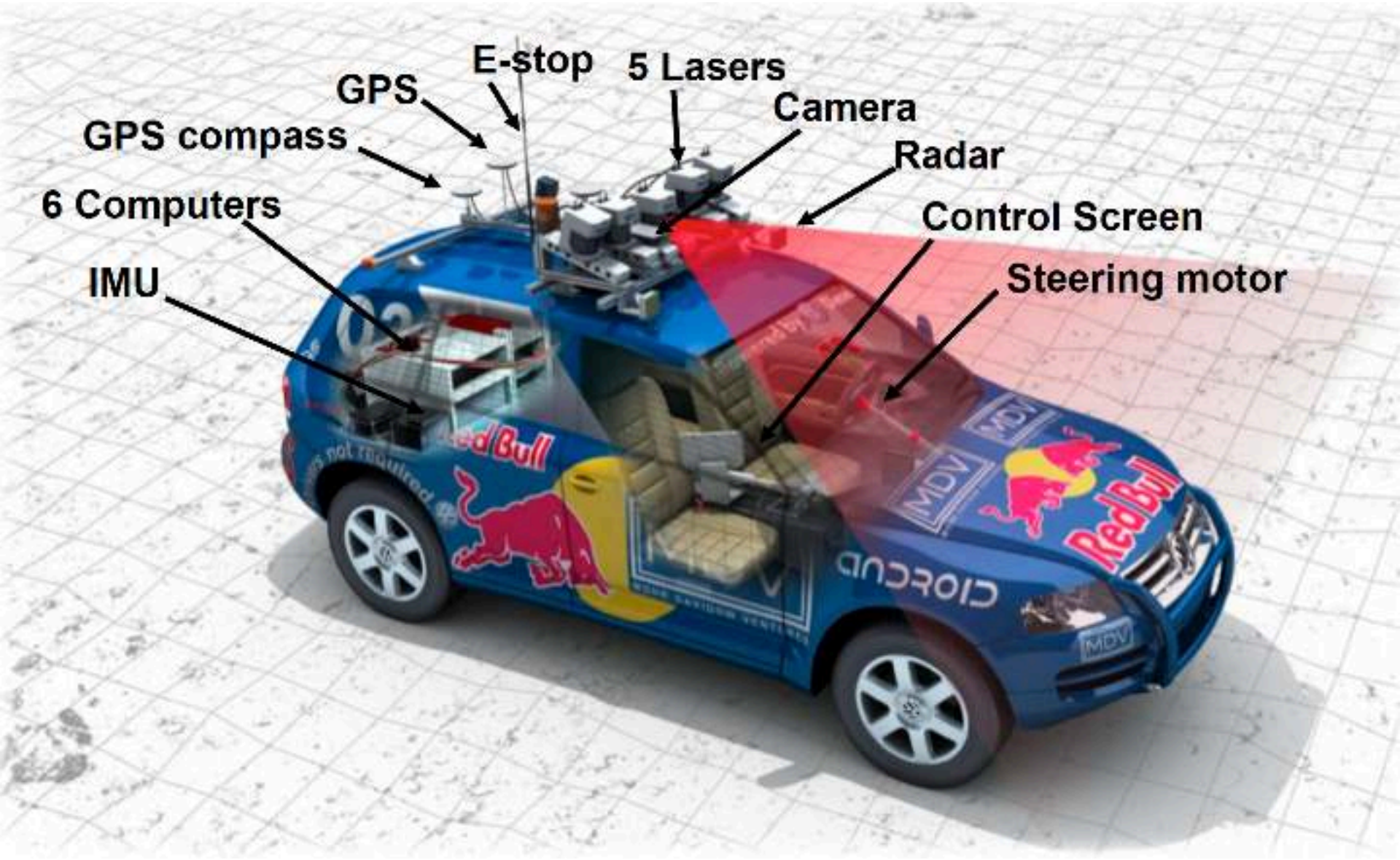
Kinect hybrid
(Microsoft)

Processing by kinect software (Microsoft)



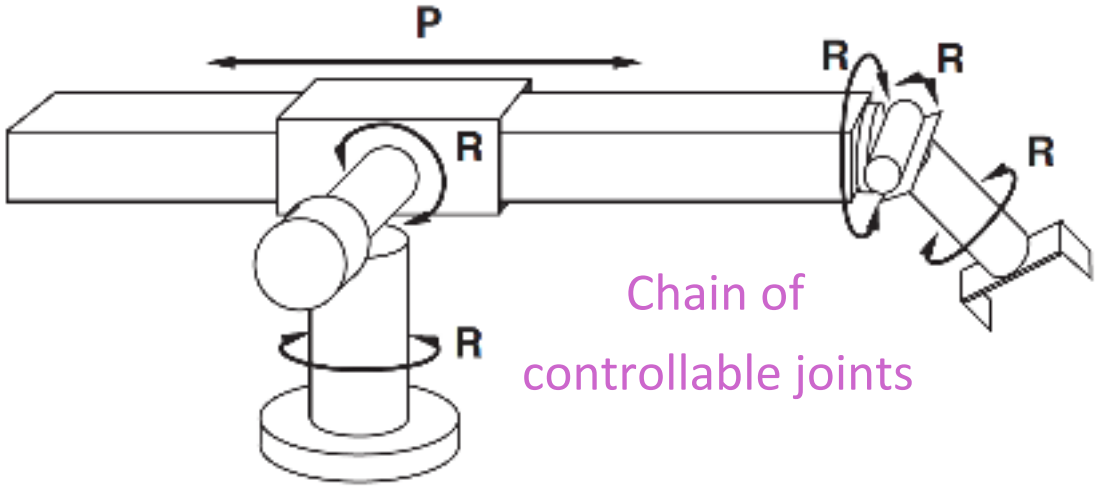
Stanley: Processing stages of computer vision: (a) Raw image, (b) Processed with the laser quadrilateral, (c) Pixel classification before thresholding, (d) Horizon detection for sky removal (Thrun et al)

Sensors


E-stop
5 Lasers
GPS
Camera
GPS compass
Radar
6 Computers
Control Screen
IMU
Steering motor
Red Bull
MDV
android

Actuators (Effectors)

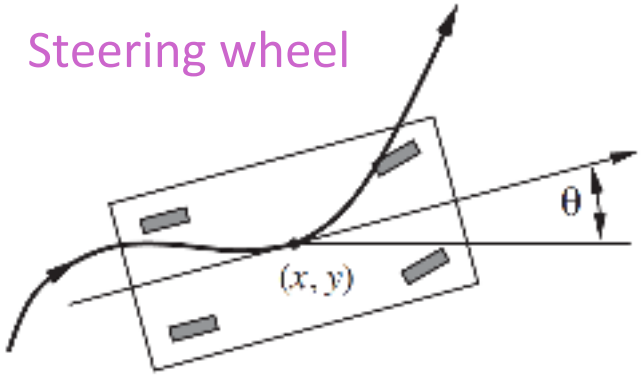
Actuators are the means used to act on the environment



Chain of controllable joints

Gas pedal (fwd/bkwd)

Steering wheel



Configuration of manipulator specified by 6 numbers

- DOF: One for each independent moving direction (6 DOFs)
- Minimal number required to position end-effector arbitrary
- Dynamical systems: Add velocity for each DOF

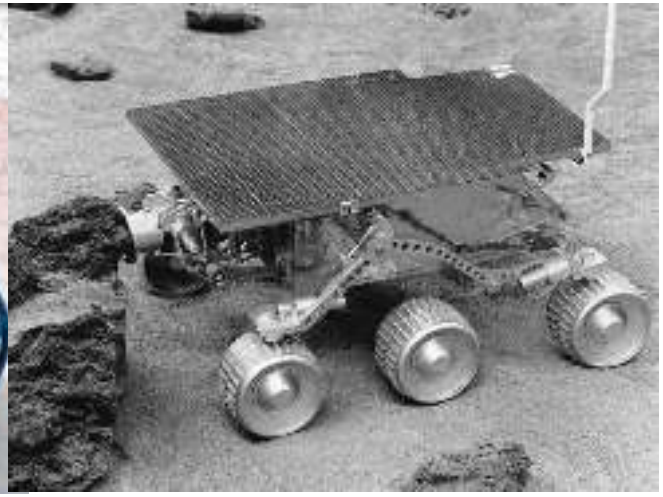
Actuators (Effectors)

Mobile robots have a range of moving mechanisms

- Including: Wheels, tracks and legs
- Differential drive: Two independently actuated wheels (tracks)
- Synchro drive: Each wheel can move/turn around its axis

Wheels

Differential drive

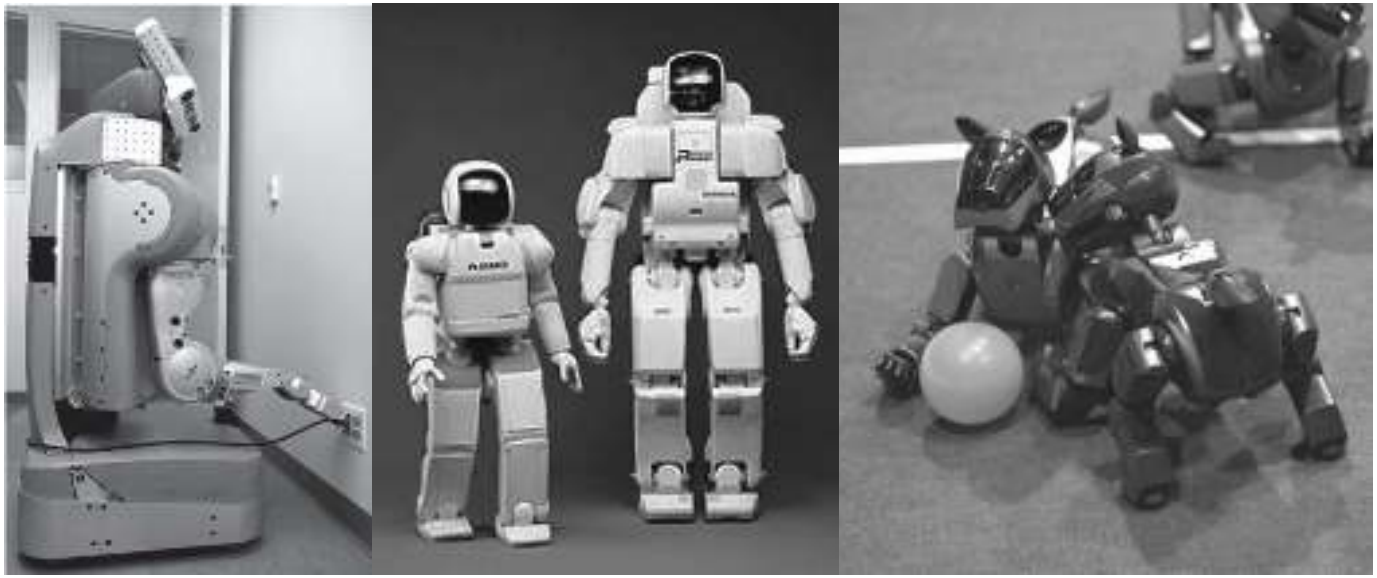


Synchro drive

Actuators (Effectors)

Some mobile robots posses arms

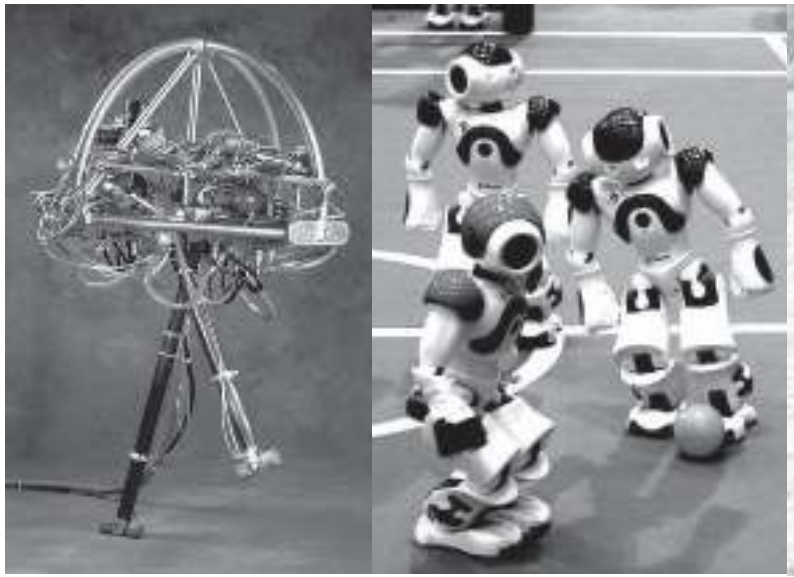
- Springs: To compensate for gravity & provide minimal resistance
- Minimize danger: for people stumbling in such robots



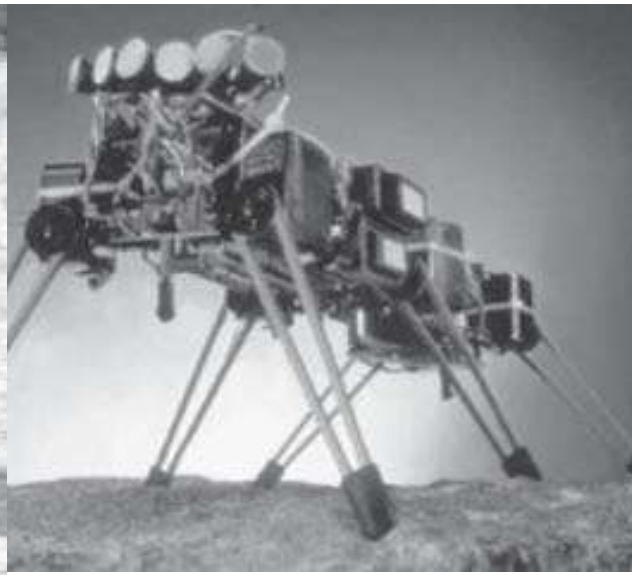
Actuators (Effectors)

Legs in contrast to wheels, can handle rough terrain

- Slow: Legs are notoriously slow on flat surfaces
- Difficult: Legs are mechanically difficult to build (1-10 legs)
- Dynamically stable: If can remain upright while hopping around
- Statically stable: If it can remain upright without moving



All these mobile robots are dynamically stable



Statically stable

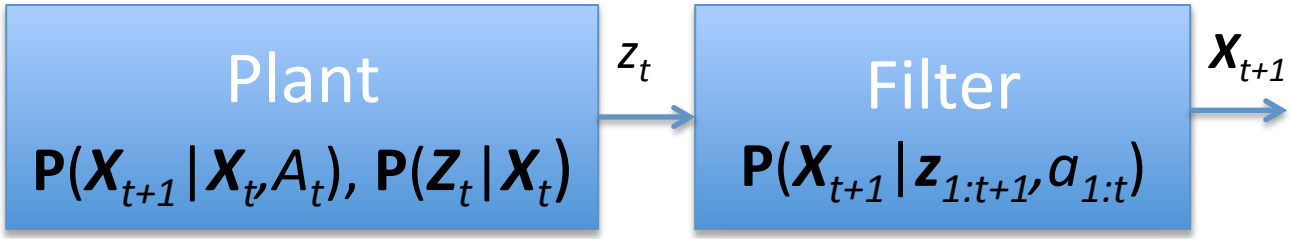
Robotic Perception

Mapping sensor measurements into internal representation

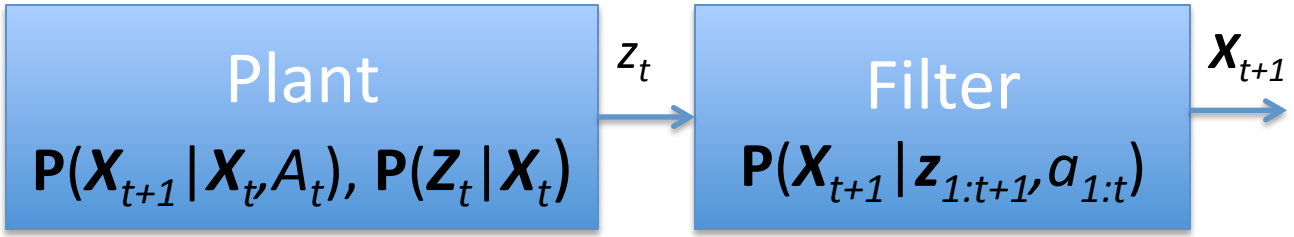
- Difficult: Sensors noisy and environment partially observable
- Filtering: Robots have all the problems of state estimation

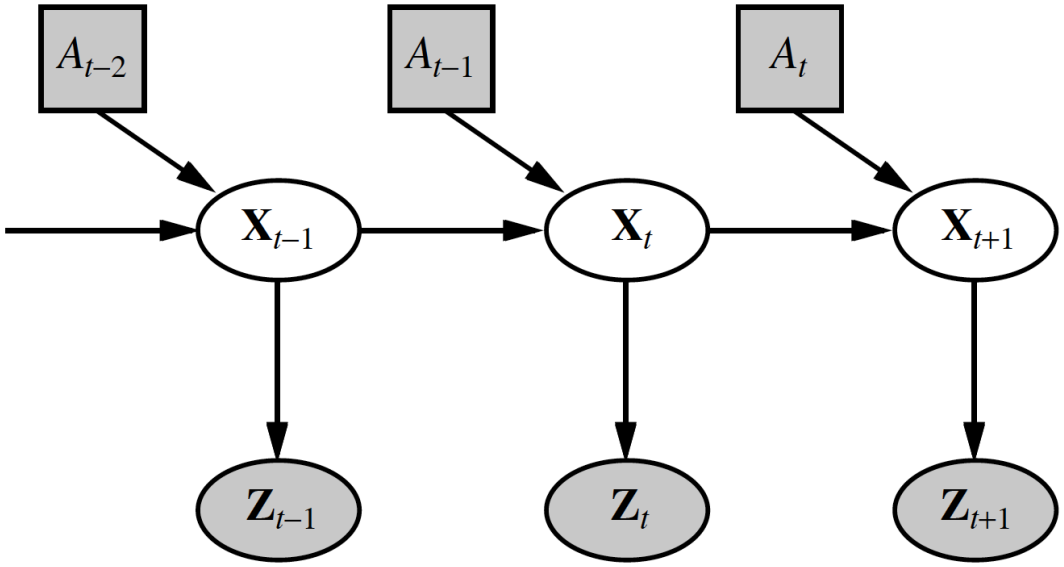
Good internal representations have three properties

- Enough information: Is contained to make good decisions
- Structured information: Helps to update it efficiently
- Natural information: Internal RVs correspond to physical world



Robotic Perception





Actions: We also condition on actions

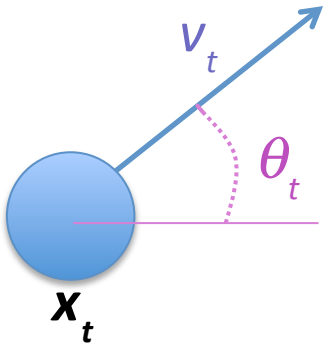
Variables: Are in this case continuous

$$P(X_{t+1}|z_{1:t+1},a_{1:t}) = \alpha P(z_{t+1}|X_{t+1}) \int P(X_{t+1}|x_t,a_t)\, P(x_t|z_{1:t},a_{1:t-1})\, dx_t$$

Localization: Where Am I?

Estimate current location and orientation (or pose)

- Example: Mobile robot moving *slowly* in a flat 2D world
- Assumption: The robot is given an exact map of environment
- Pose: Defined by x,y and heading angle θ. Hence: $\boldsymbol{X}_t = (x_t, y_t, \theta_t)^t$

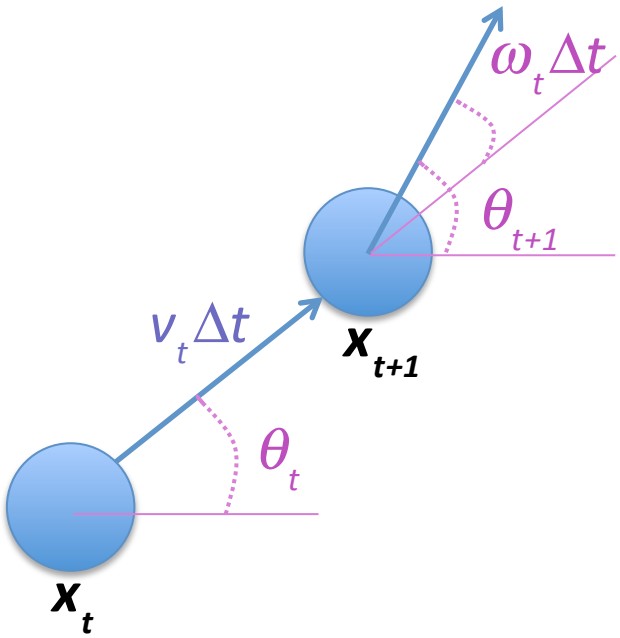


Localization: Where Am I?

An action is the instantaneous specification of v_t and ω_t

$$\hat{\boldsymbol{X}}_{t+1} = f(\boldsymbol{X}_t, v_t, \omega_t)^{\mathrm{t}} = \boldsymbol{X}_t + \begin{pmatrix} v_t \Delta t \cos\theta_t \\ v_t \Delta t \sin\theta_t \\ \omega_t \Delta t \end{pmatrix}$$ (deterministic prediction)

$$\mathbf{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{X}_t, \mathrm{v}_t, \omega_t) = \mathrm{N}(\hat{\boldsymbol{X}}_{t+1}, \boldsymbol{\Sigma}_x)$$ (Gaussian: mean $\hat{\boldsymbol{X}}_{t+1}$, covariance $\boldsymbol{\Sigma}_x$)

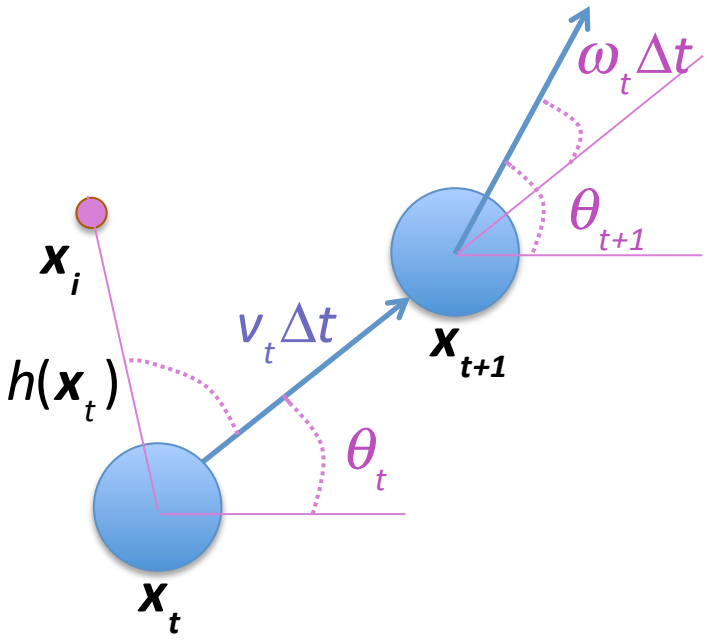


Localization: Where Am I?

Sensors detect stable, recognizable env features (landmarks)

$$\hat{\boldsymbol{z}}_t = h_t(\boldsymbol{x}_t) = \begin{pmatrix} \sqrt{(x_t - x_i)^2 + (y_t - y_i)^2} \\ \arctan(y_t - y_i)/(x_t - x_i) - \theta_t \end{pmatrix}$$ (deterministic)

$$\mathbf{P}(\boldsymbol{z}_t \mid \boldsymbol{x}_t) = \mathrm{N}(\hat{\boldsymbol{z}}_t, \boldsymbol{\Sigma}_z)$$ (Gaussian)



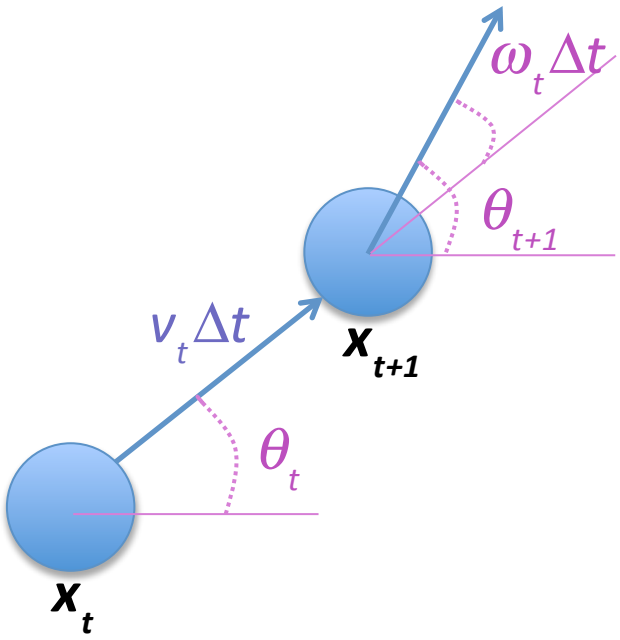
Localization: Where Am I?

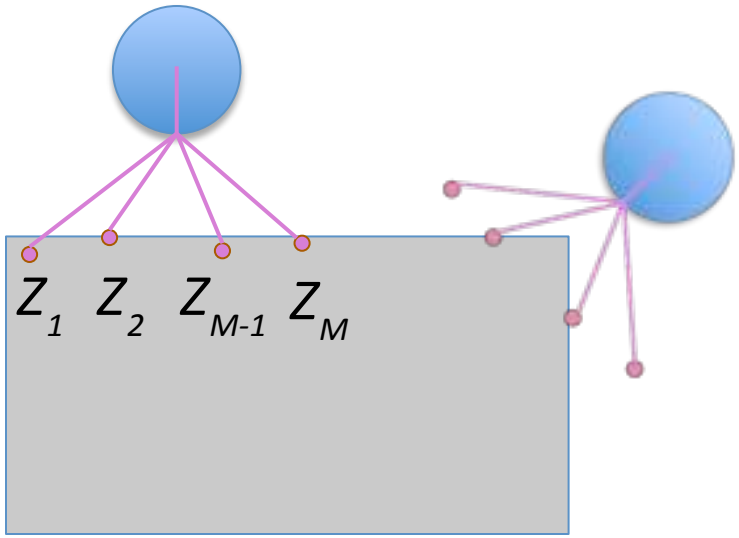
Array of M range sensors with fixed bearing relative to robot

- Gaussian beam errors: Independent and identically distributed

$$P(z_t \mid x_t) = \alpha \prod_{j=1}^{M} e^{(z_t - z_j)/2\sigma^2} \quad \text{(Gaussian)}$$

- Advantage: No landmark identification necessary before scan





Localization: With Particle Filtering

function MC-Localization ($v, \omega, N, P(X'|X,v,\omega), P(z|z^*), map$) returns S

persitent S, S' // vector of samples of size N

local W' // temporary vector of weights of size N

if ($S = \varnothing$) for ($i = 1{:}N$) $S[i] = \text{Sample}(P(X_0))$ // intializat phase

for $i = 1{:}N$ { // update cycle

$S'[i] = \text{Sample}(P(X' \mid X = S[i], v, \omega))$ // Step 1

$W'[i] = \mathbf{1}$ // Step 2

for ($j = 1{:}M$) {

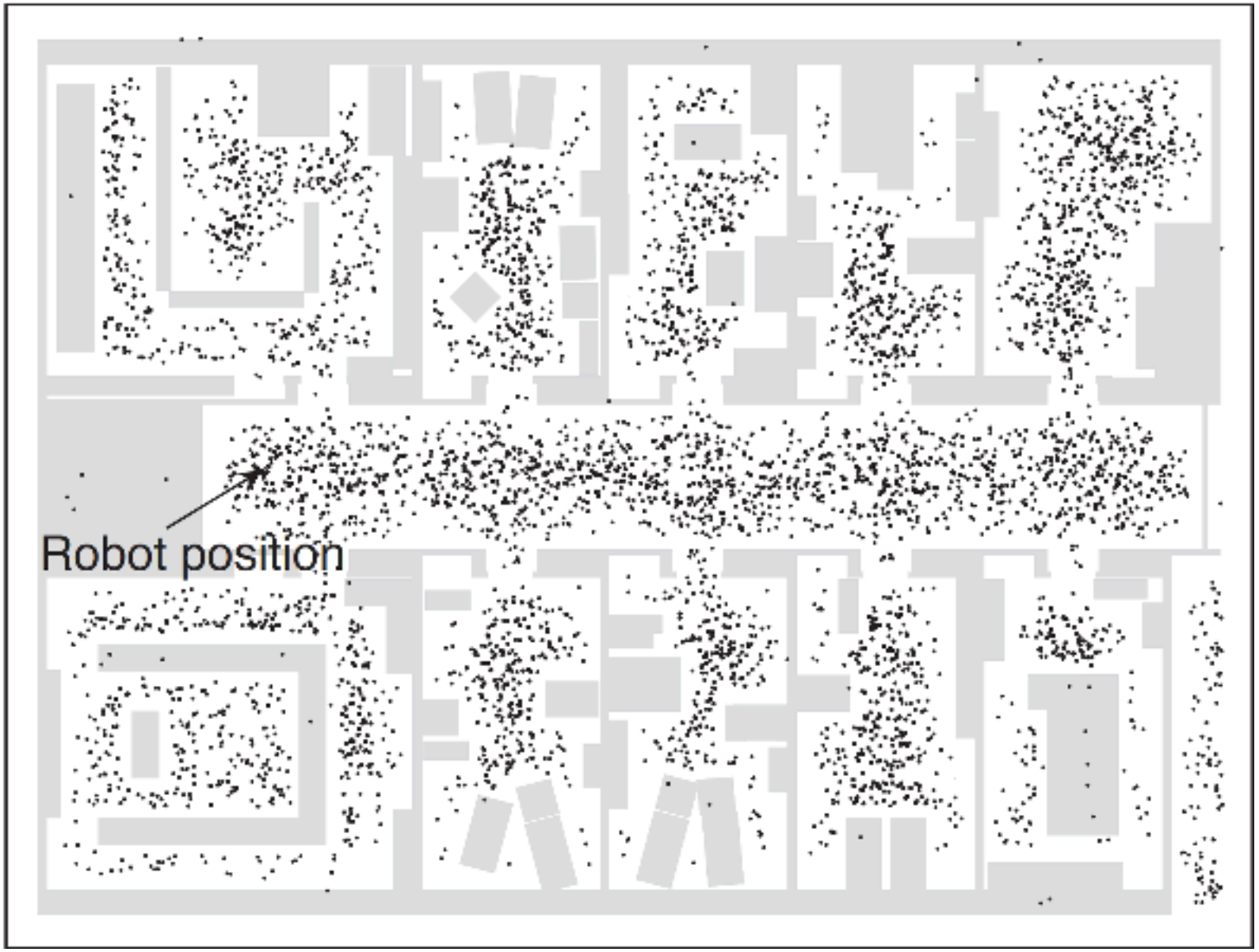
$z^* = \text{RayCast}(j, X = S'[i], map)$ // get beam

$W'[i] = W'[i] \cdot P(z_j \mid z^*)$ } } // adjust weight

$S = \text{Weighted-Sample-With-Replacement}(N, S', W')$ // Step 3

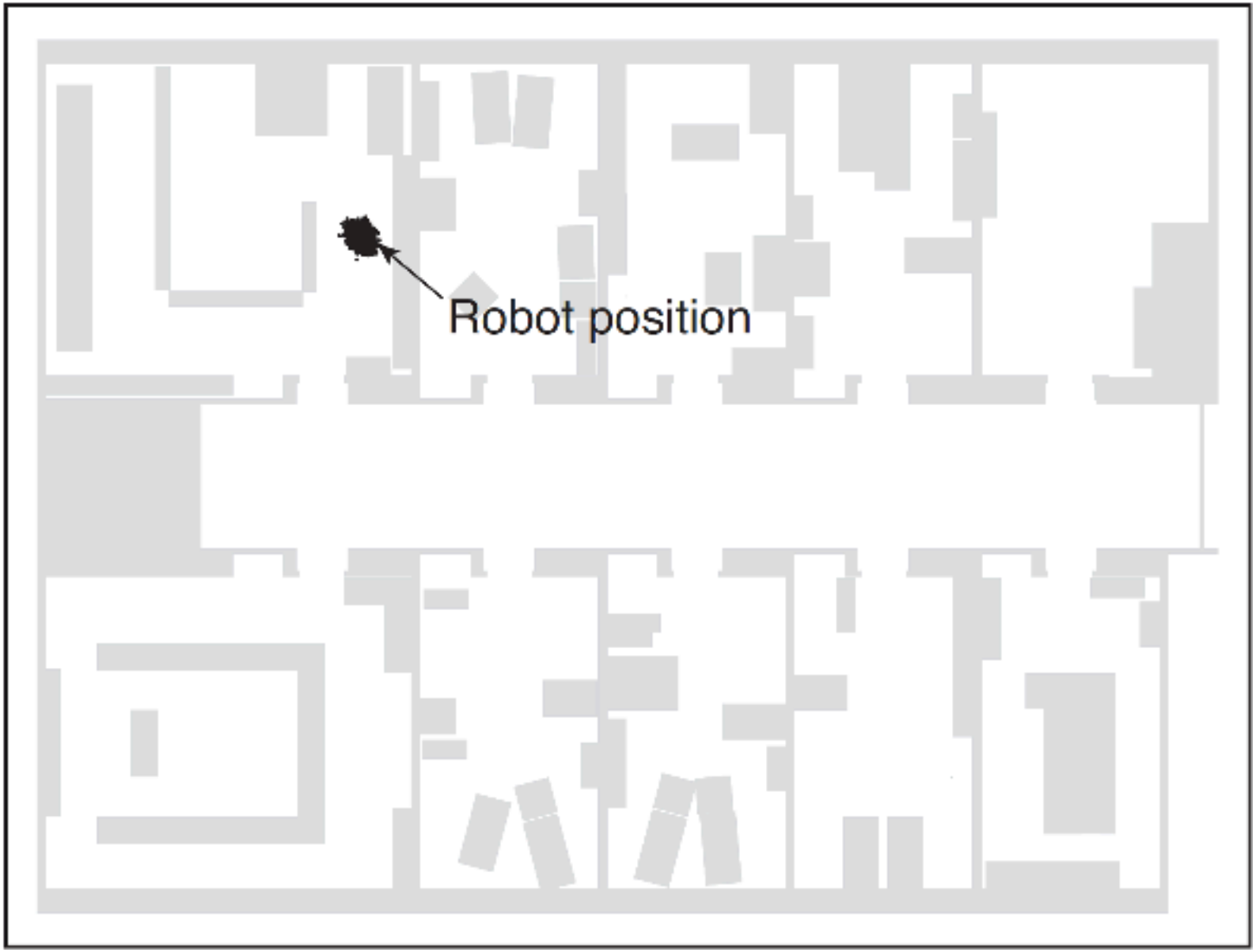
return S

Localization: With Particle Filtering



Localization: With Particle Filtering

Localization: With Particle Filtering



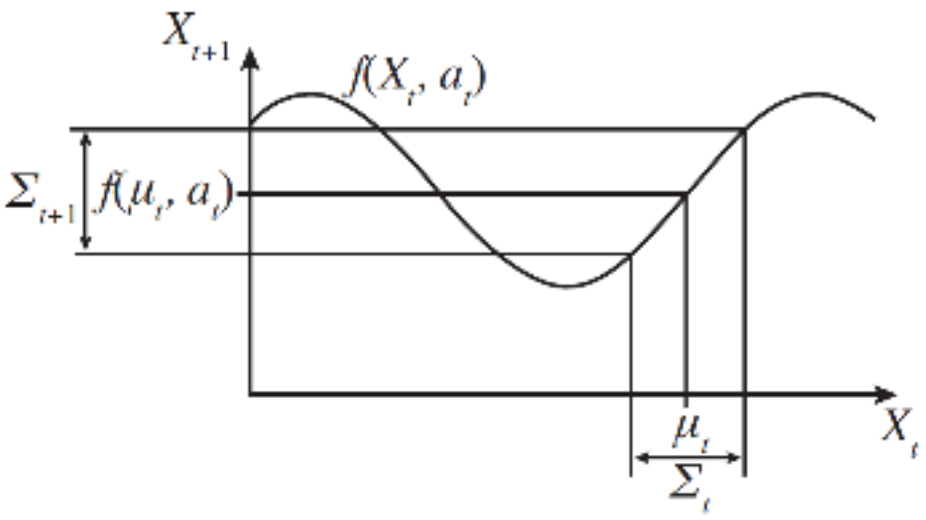
Localization with Kalman Filtering (KF)

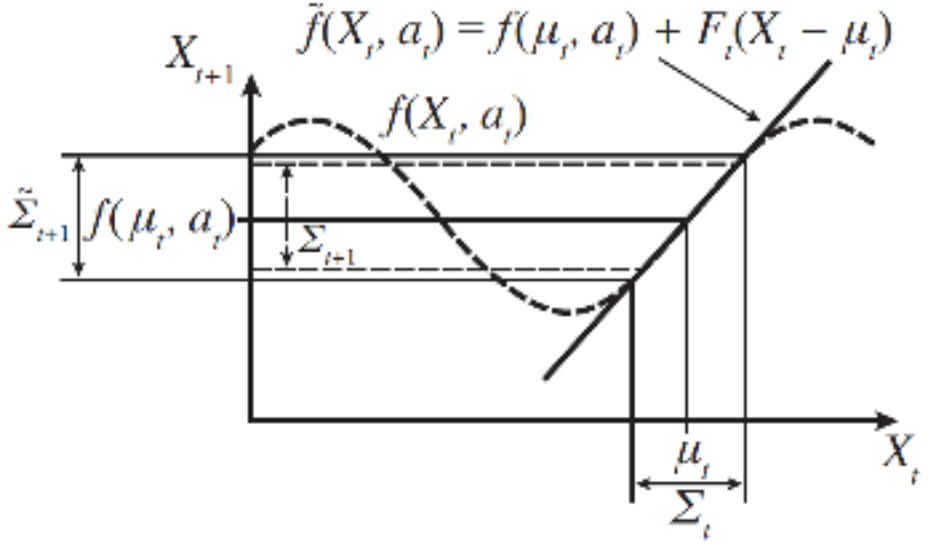
A Kalman filter represents $\mathbf{P}(\boldsymbol{X}_t | \boldsymbol{z}_{1:t}, a_{1:t-1})$ by a Gaussian

- Mean is denoted by $\boldsymbol{\mu}_t$ and covariance is denoted by $\boldsymbol{\Sigma}_t$
- Problem: Closed only under linear *motion f* and *measurement h*

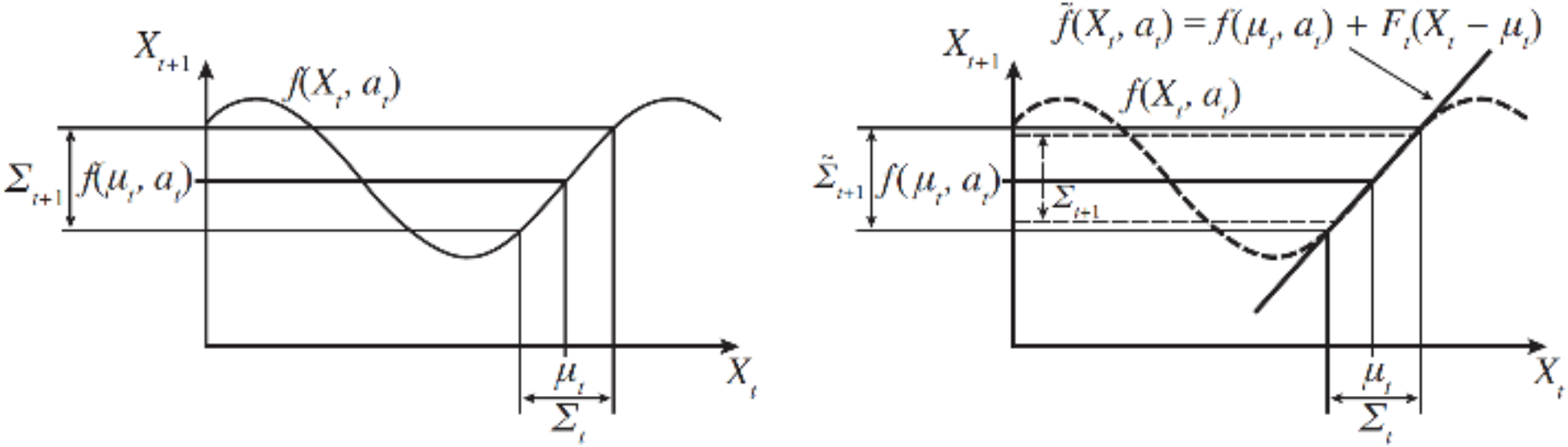
In nonlinear motion/measurements update is non Gaussian

- Fix: Linearize the motion and sensor models
- Local approximation of a nonlinear function with linear one

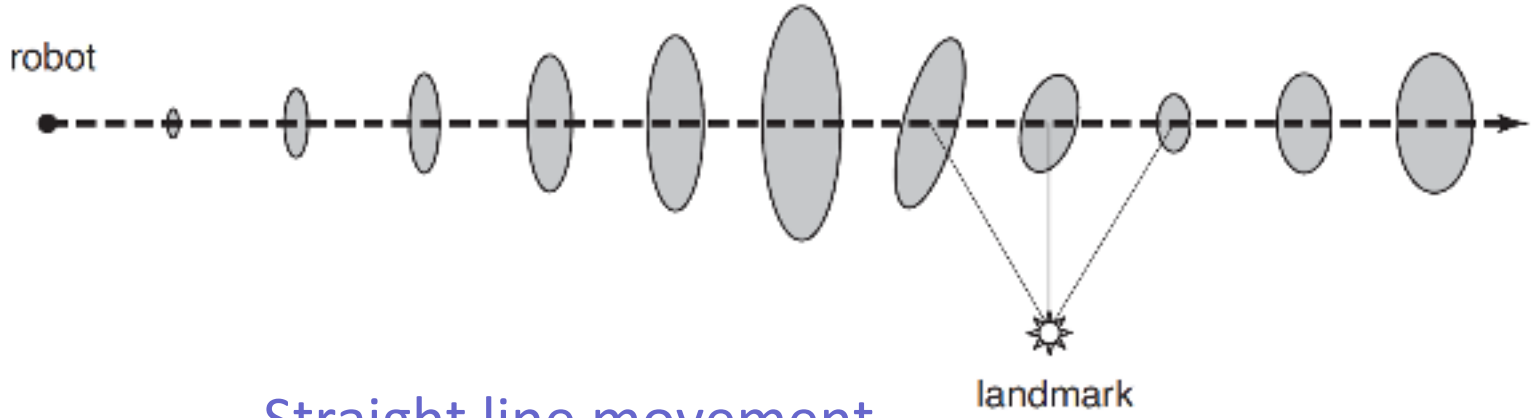




Localization with Extended KF (EKF)



One dimensional illustration of a linearized model



Straight line movement

Uncertainty increases gradually until observes landmark with known position

Mapping

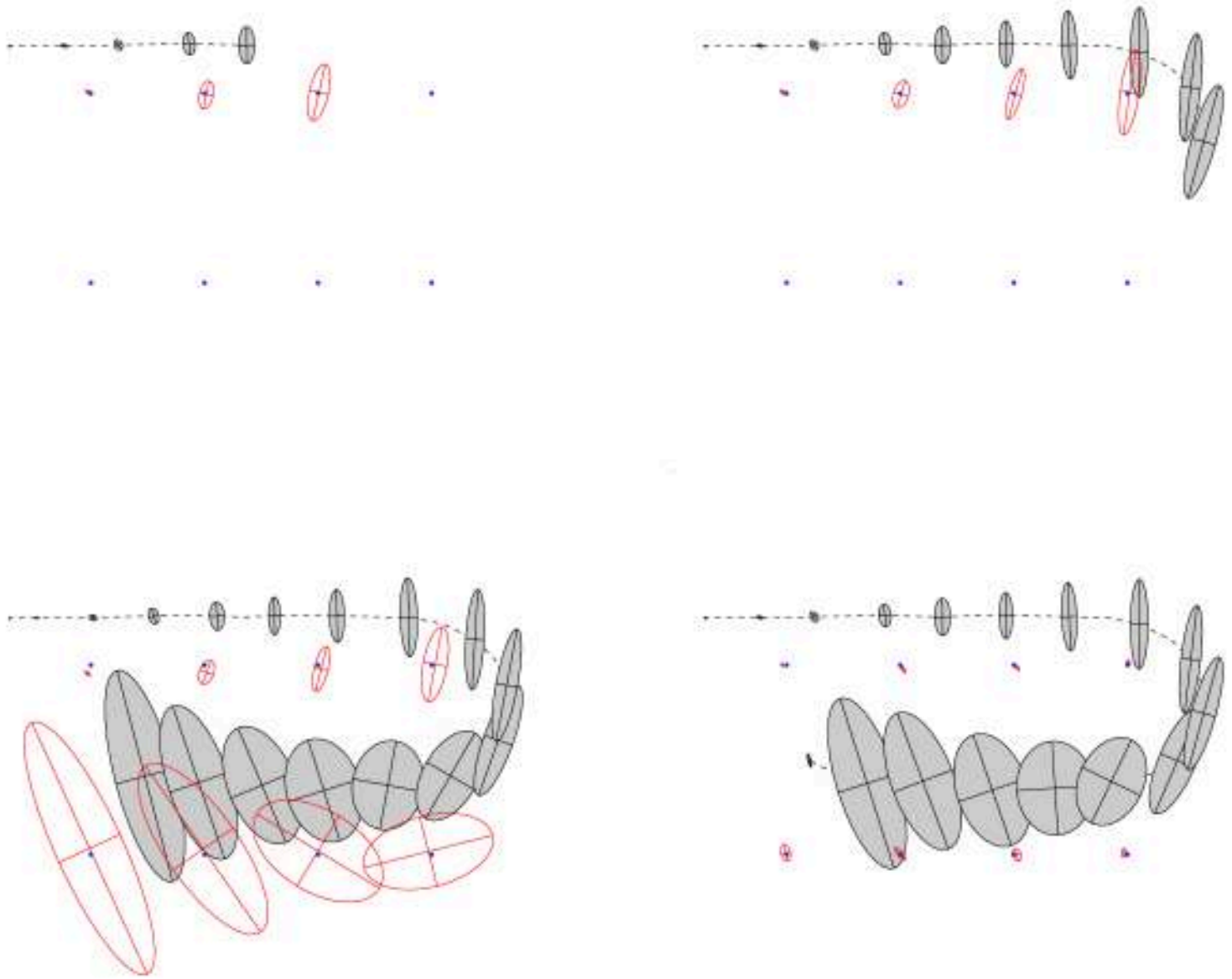
In some situations no map of environment is available

- Robot will have to acquire a map
- Chicken: Determine location relative to an unknown map
- Egg: Construct map while doesn't quite know location
- SLAM: simultaneous localization and mapping

Solved using probabilistic techniques. For EKF pretty easy

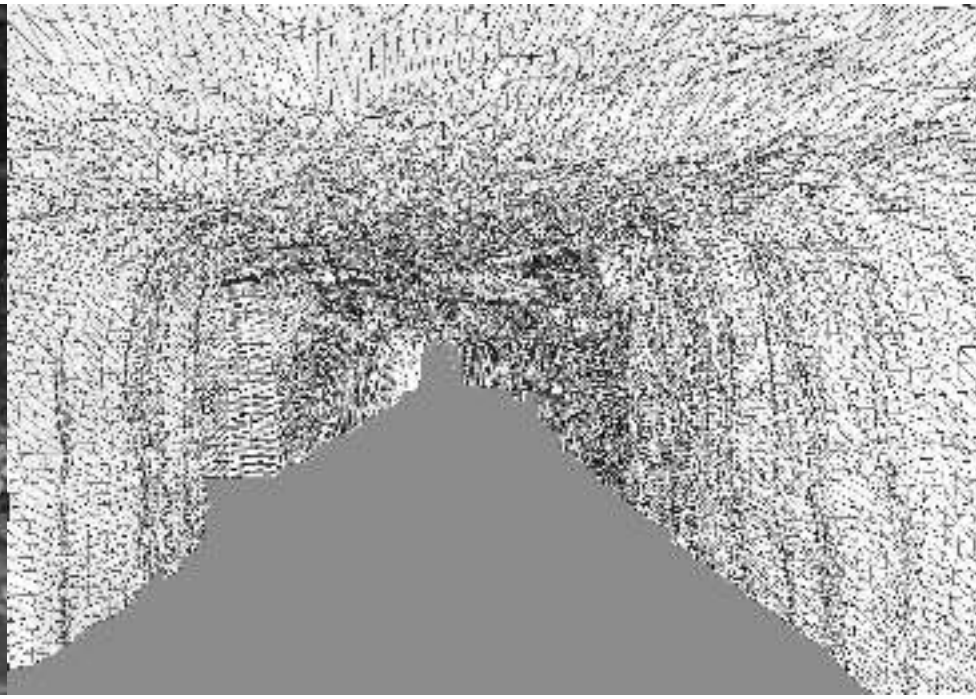
- Augment state vector to include locations of landmarks in env
- Proceed as done before for localization
- EKF update scales quadratically so for small maps SLAM feasible

Mapping



Simultaneous Localization & Mapping

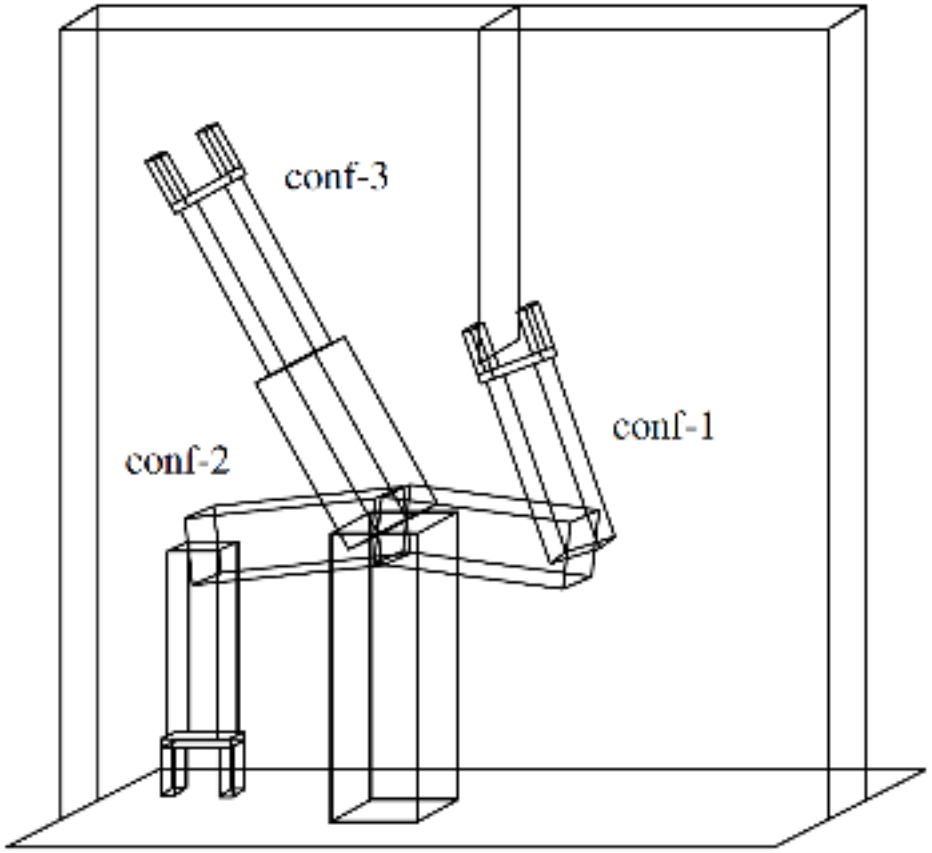
Robot mapping and abandoned coal mine

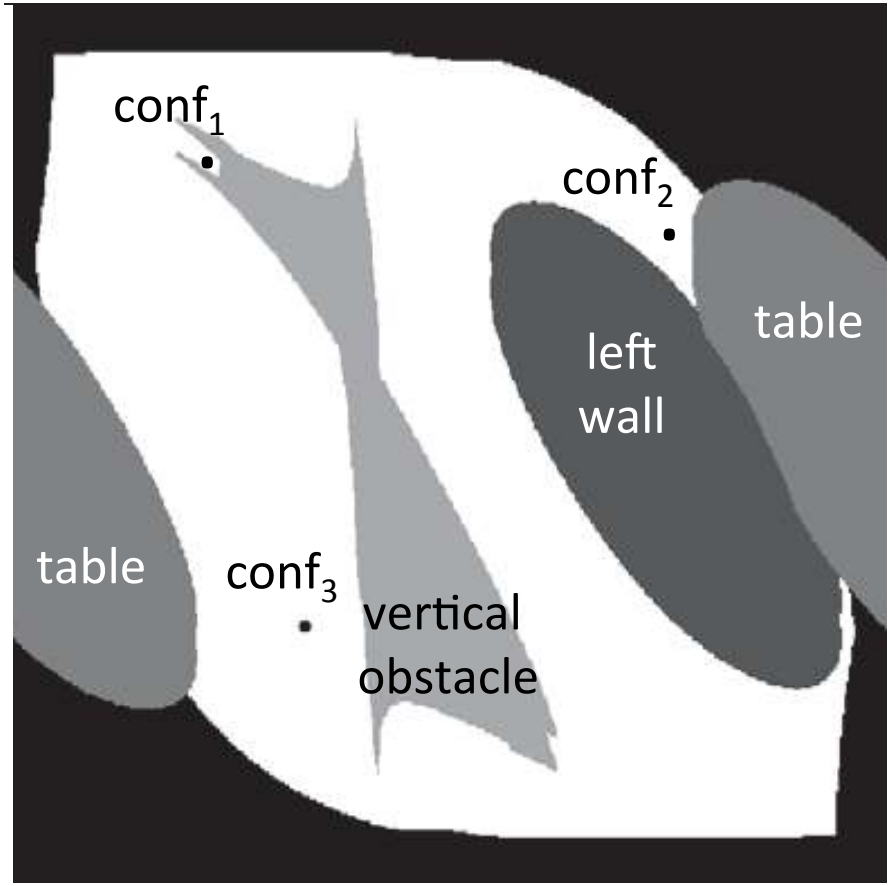


3D map of the mine acquired by the robot

Motion Planning

Idea: Plan in *configuration space* defined by robot's DOFs





Solution: Is a point trajectory in free configuration space

Configuration-Space Planning

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

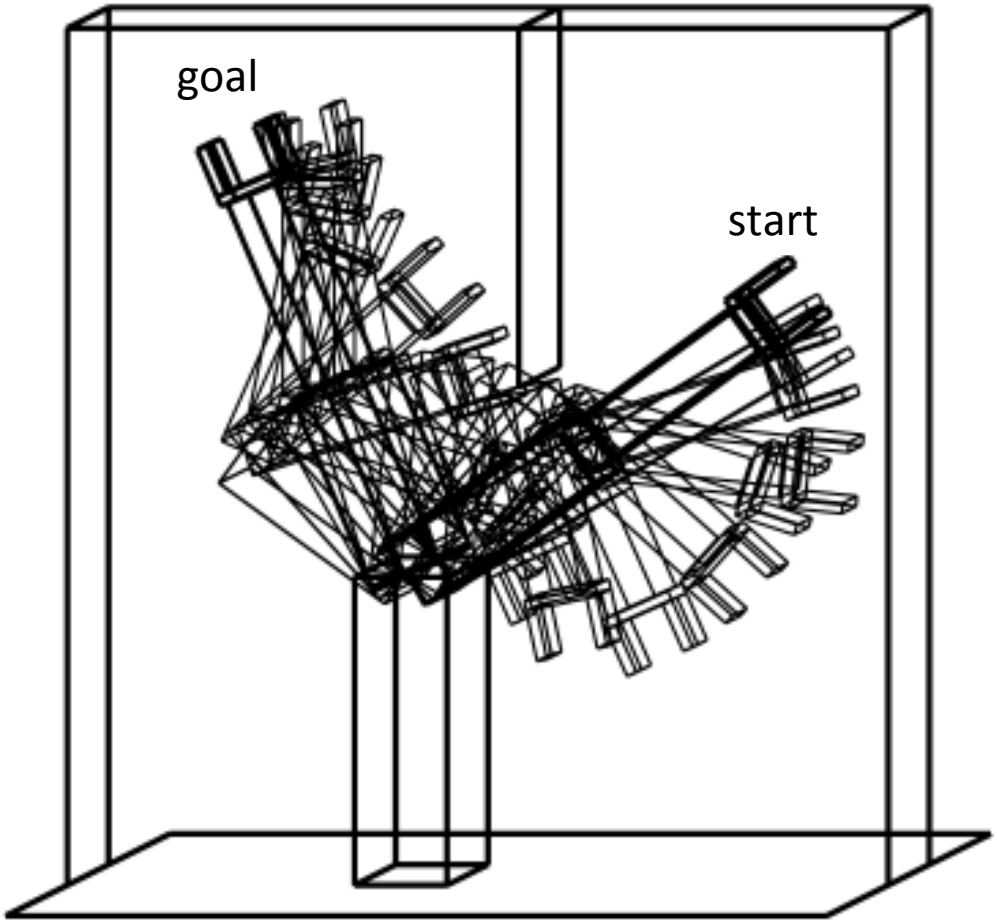
Cell decomposition:

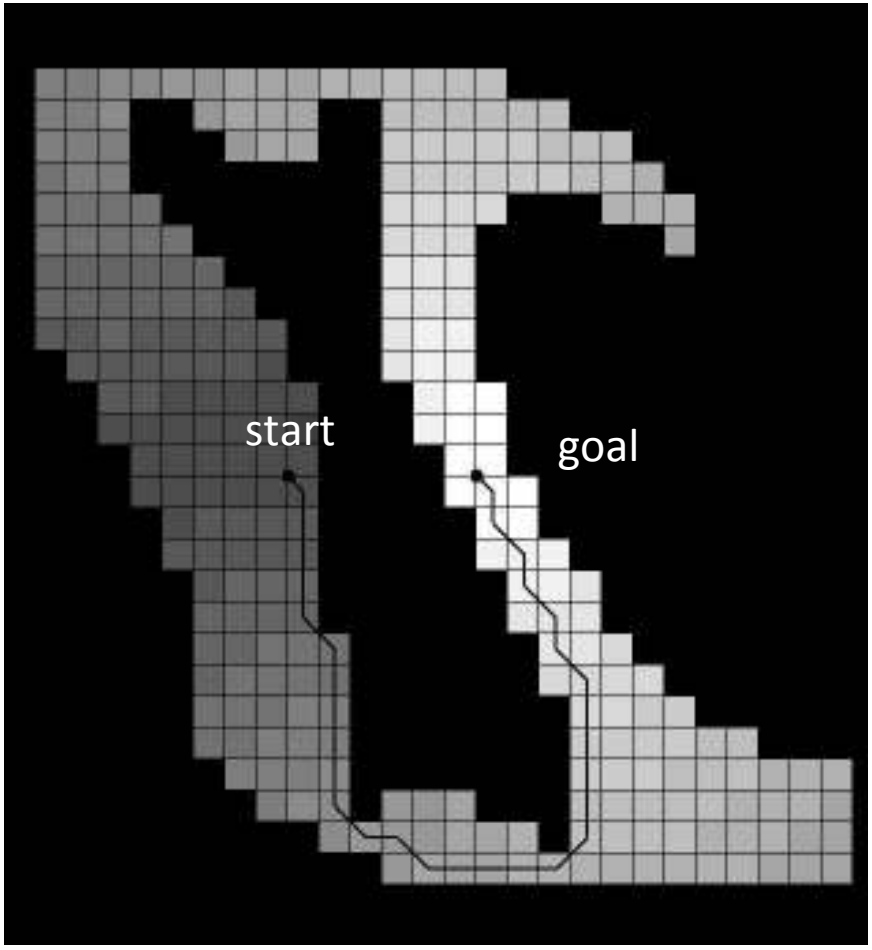
- Divide up space into simple cells
- Each of which "easily" traversed (e.g., convex)

Skeletonization:

- Identify finite number of easily connected points/lines
- That form a graph such that
- Any two points are connected by a path in the graph

Cell-Decomposition Example

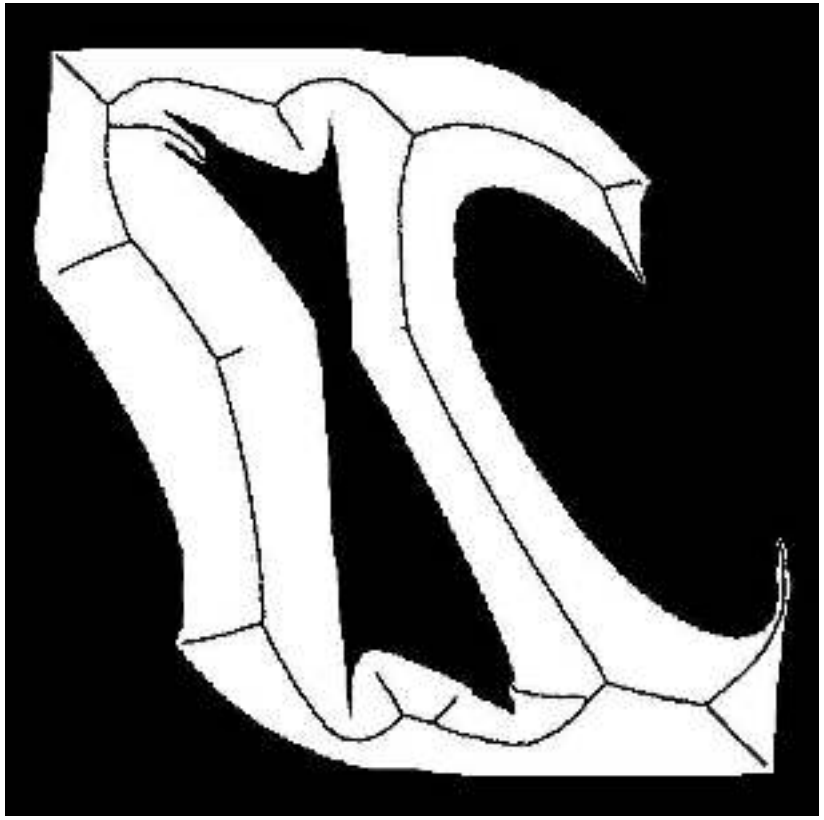




- Problem: There may be no path in pure free-space cells
- Solution: recursive decomposition of mixed (free+obstacle) cells

Skeletonization: Voronoi Diagram

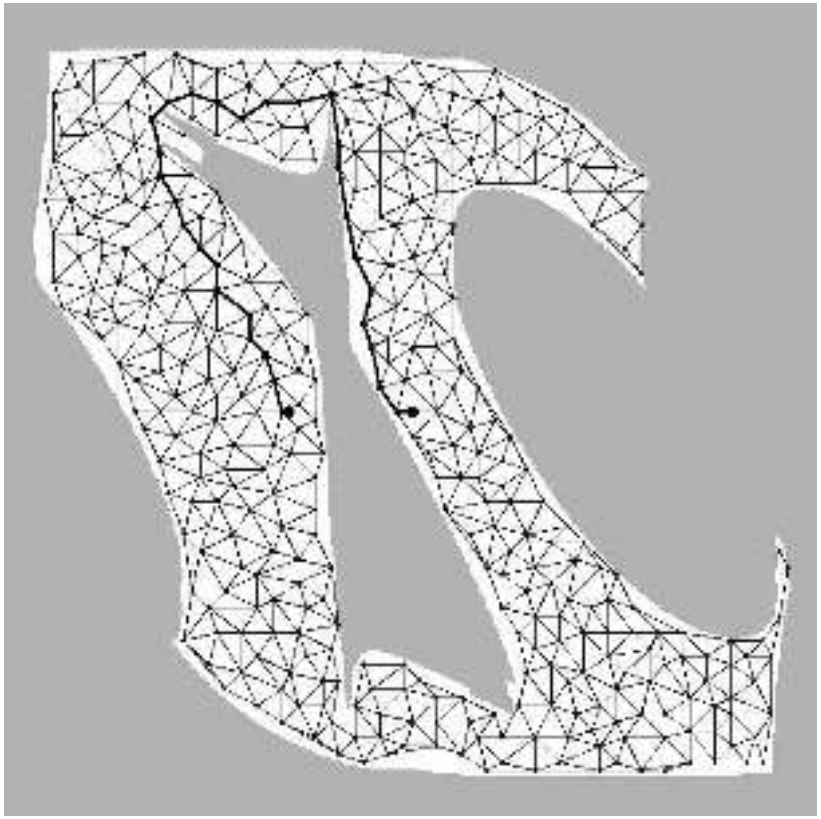
- Voronoi diagram: Locus of points equidistant from obstacles



- Problem: doesn't scale well to higher dimensions

Skeletonization: Probabilistic Roadmap

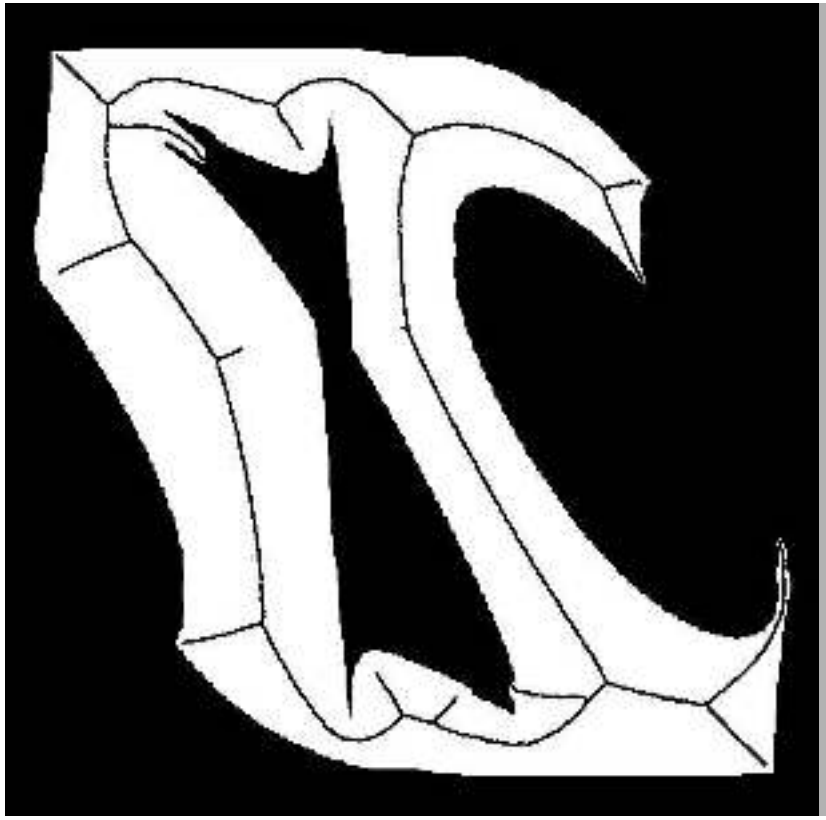
- Generate random points in the configuration space
- Keeping those in free space create graph by joining pairs by lines



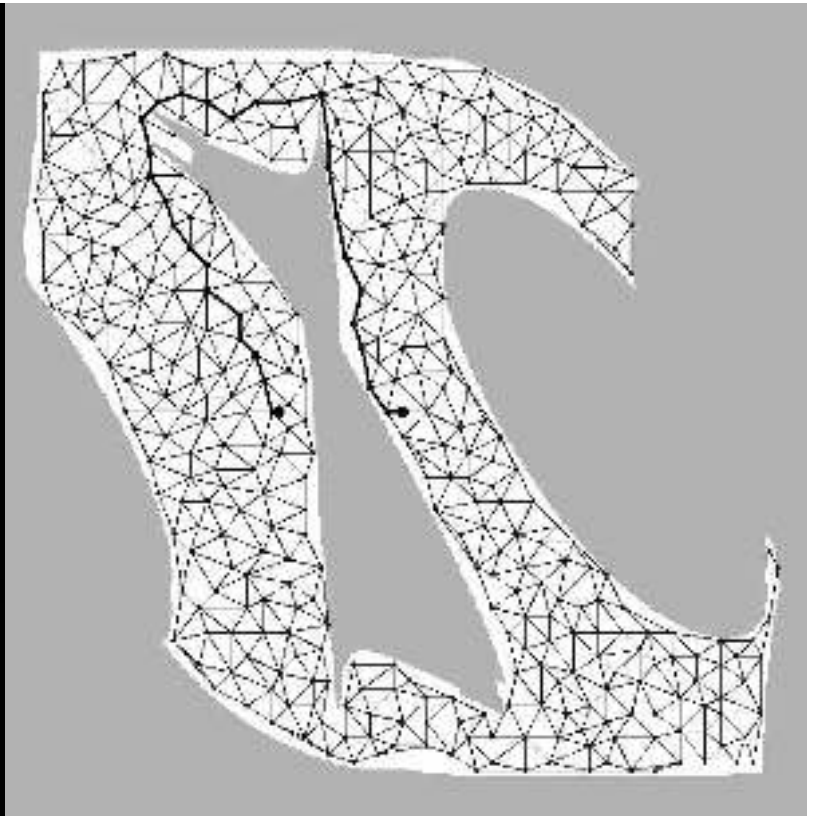
- Problem: need to generate enough points to ensure that
- Every start/goal pair is connected through the graph

Skeletonization

Voronoi Diagram



Probabilistic Roadmap



Moving

The plans were produced by deterministic planners

- Assume: That the robot can simply follow any path in C-space
- In real world: This is not the case eg. because of inertia
- Robots get to exert forces: Rather than specifying positions

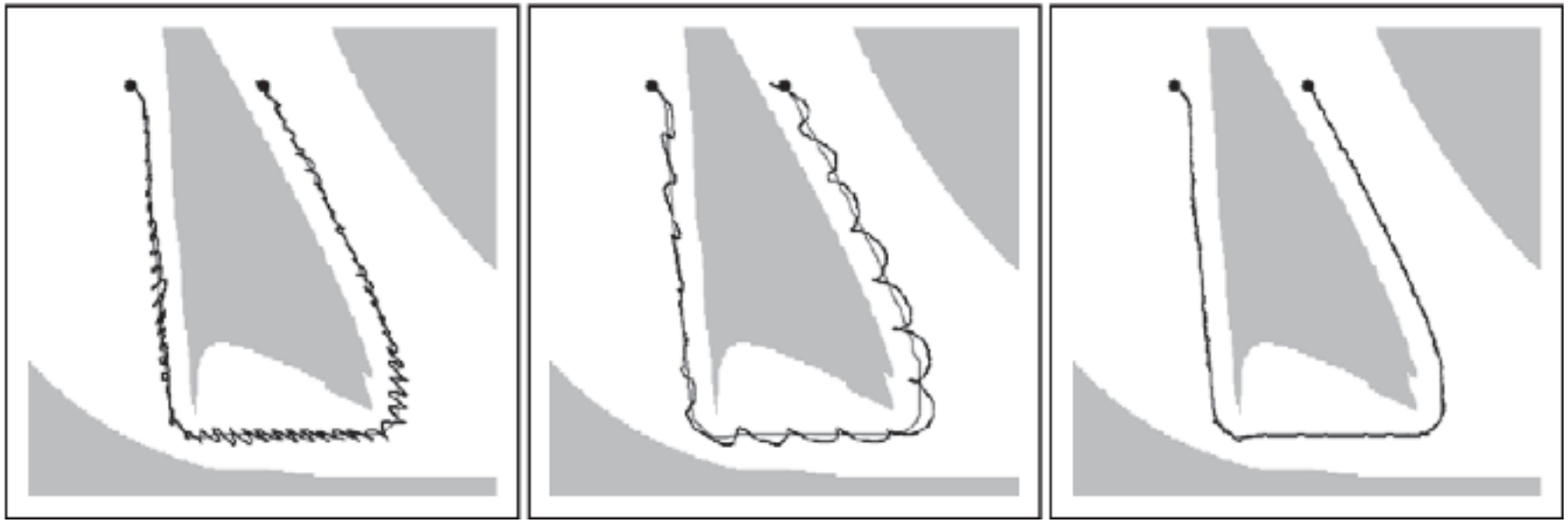
Dynamic states extends kinematic state by its velocity

- In addition to position-angle: Add their rate of change (forces)
- Differential equations: Typically used to express such models
- Advantage: Leads to superior robot performance
- Kinematic path planners: Used for practical purposes

Control

Controllers compensate for path-planning limitations

- Reference controller: Keeps the robot on the reference path
- Optimal controller: Optimize a global cost function (policies)
- Stable controller: Small perturbation lead to bounded error
- Strictly stable: Able to return and stay on its reference path



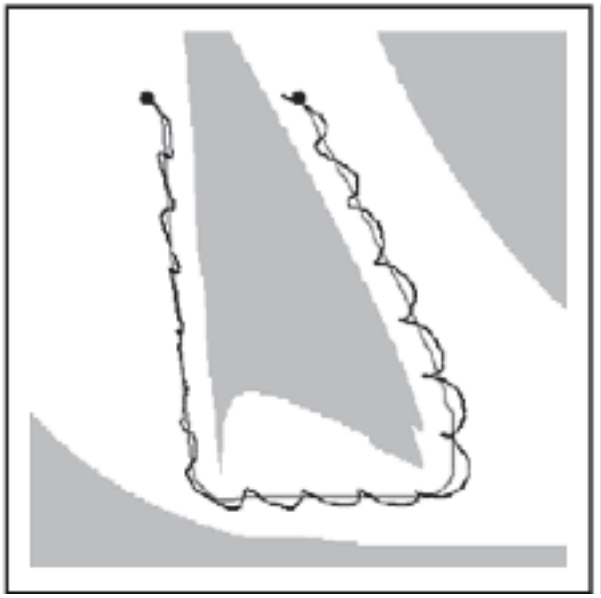
Reference path and the jitter introduced by various controllers

PID Control

Compensate errors to reference path with counterforces:

- Proportional: $a(t) = K_P(y(t) - x(t))$ spring
- Derivative: $a(t) = K_D \partial(y(t) - x(t)) / \partial t$ damping
- Integrative: $a(t) = K_I \int_{-\infty}^{t} (y(\tau) - x(\tau)) d\tau$ persistent error

P: $K_P = 1$



P: $K_P = 0.1$

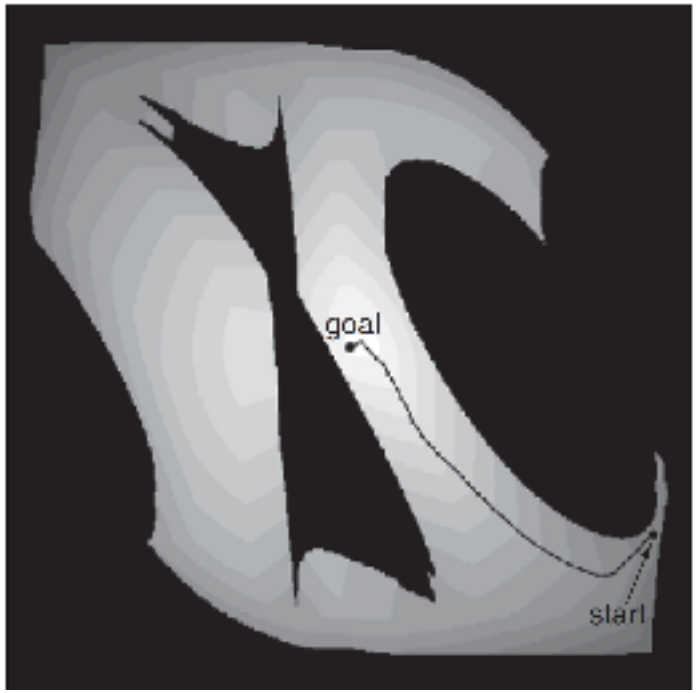
PD: $K_P = 0.3,\ K_D = 0.8$

Potential-Field Control

Generate robot motion directly and drop path planning:

- Attractive field: Pulls the robot toward its goal
- Repelling field: Pushes robot away from obstacles
- Global minimum: The goal configuration

Successful path



Local optimum

$$|g - x(t)| + \int |o - x(t)|\, dx$$

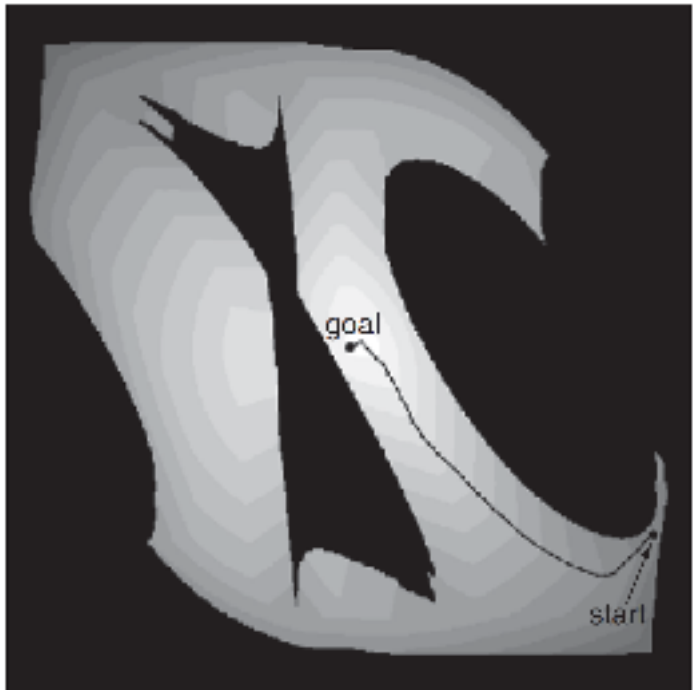
no velocity

Potential-Field Control

Generate robot motion directly and drop path planning:

- Real time: Well suited for real time control
- Trajectory: Computed with hill-climbing techniques
- Optimization: Amounts to calculating the gradient

Successful path



Local optimum

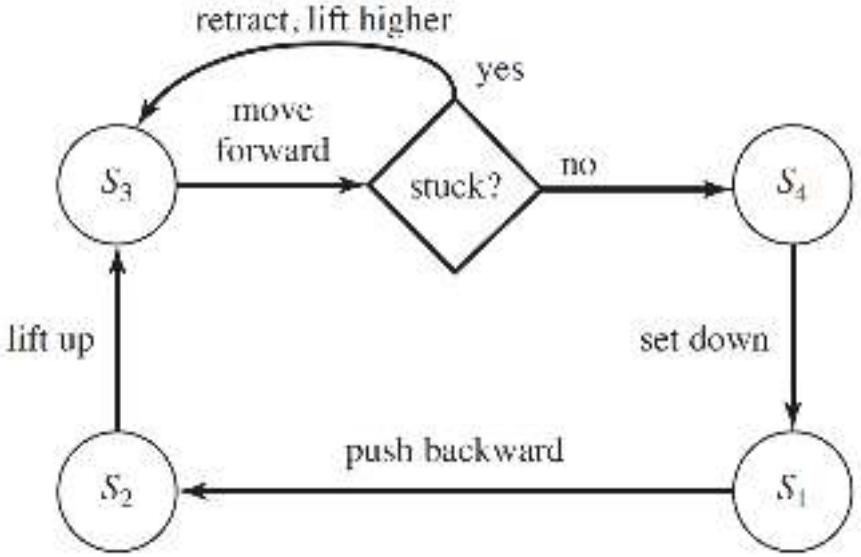
Great for local robot motion and slow robots

Reactive Control

Work without an explicit environment model:

- Models: Sufficiently accurate are difficult to obtain
- Problems: Computational difficulties and localization error
- Architecture: Reflex agent architecture

Genghis the hexapod robot



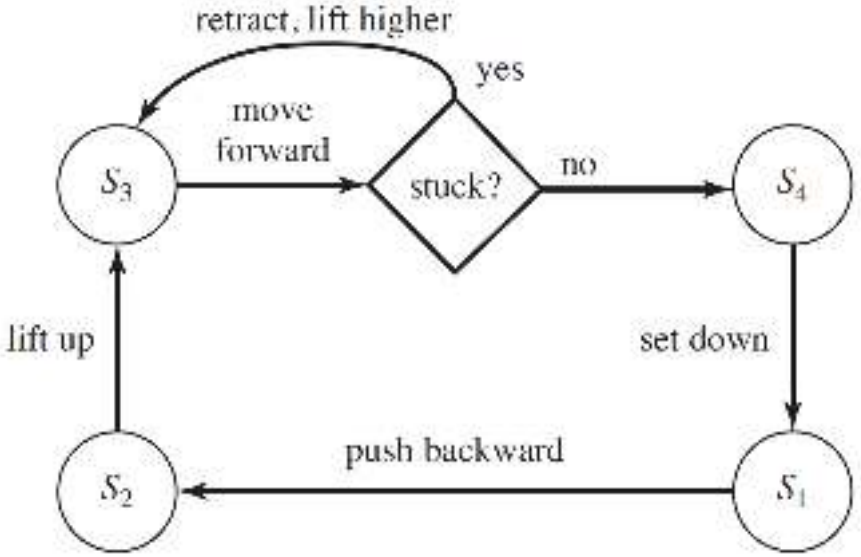
Augmented FSM for one leg

Reactive Control

Instead of a reference path use a movement pattern:

- Stable gait: Right-front, right rear, left centers, then other 3
- Works well: On flat terrains
- On rugged terrain: Use the augmented FSM

Genghis the hexapod robot



Augmented FSM for one leg

Reinforcement-Learning Control

Policy-search form of reinforcement learning:

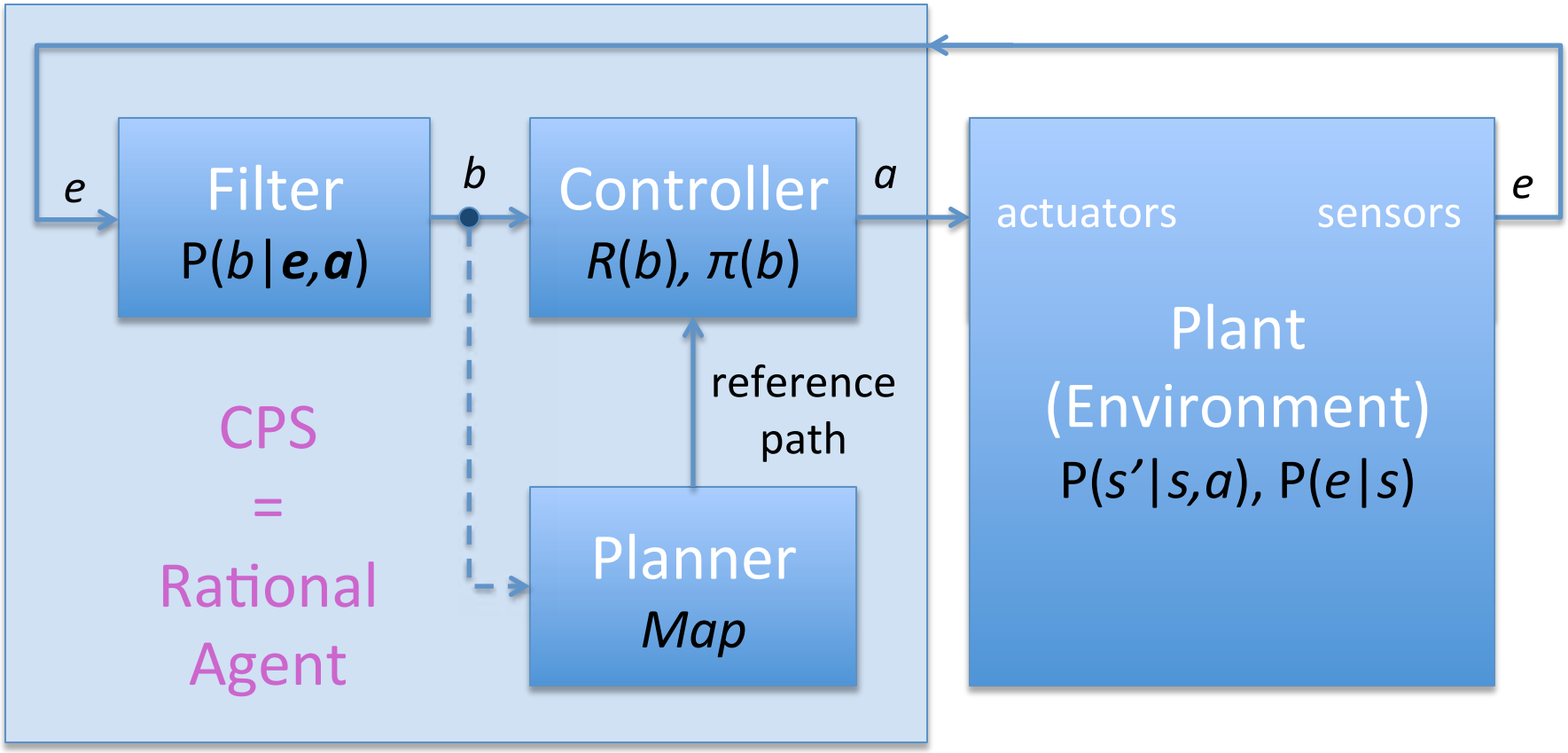
- Success: Enormously influential in recent years
- Solved: Challenging robotics problems with no solution
- Example: Autonomous acrobatic helicopter flight (flip)
- Nonlinear: Nature of the aerodynamics involved (few humans)
- Efficient: Using a few minutes policy search learned a policy

Reinforcement-Learning Control

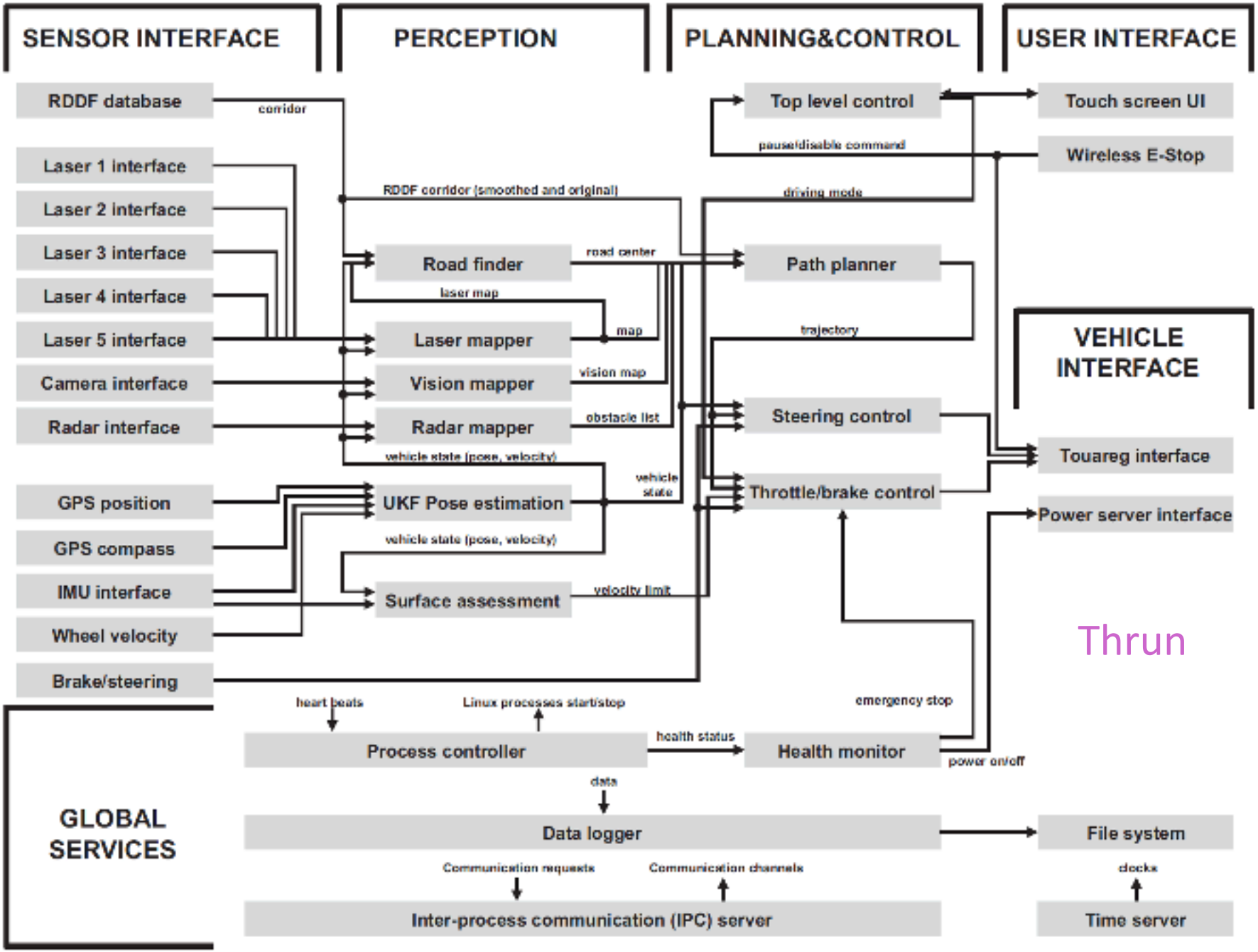
Policy-search needs an accurate model of the domain:

- Model input: State & controls (at t), resulting state (at $t+\Delta t$)
- State: 3D coordinates, yaw, pitch, roll angles, and derivatives
- Controls: Throttle, pitch, elevator, aileron, and rudder
- Getting a model: Let a human expert fly and record controls
- Learning: 4' record enough to build a predictive model

Summary: Rational Agent



Pipeline Software Architecture



Try It Out In Our Lab: Rovers

- 3 x Mobile Robots Pioneer 3-AT
  - SICK LMS 100 Laser Scanner
    - 0.5 – 20 m operating range
    - 270° field of view
  - Cannon VC-C50i PTZ Analog Camera
  - UHF RFID-Reader
  - Cyton Gamma 300 Manipulator Arm
    - 300 g payload
    - 53.4 cm total reach
  - Sonar Distance Sensors
  - Bumper Switches

Try It Out In Our Lab: Quadcopters

- 2 x AscTec Pelican Drones
  - Laser Scanner 0.06 – 4 m range
  - CMOS Camera
  - 1.6 GHz Intel Atom Processor Board
  - 2.1 GHz Intel Core i7 Quad-Core Board
  - Linux Operating System

- 3 x Parrot AR.Drone2.0
  - Front (720p) and Floor (QVGA) Camera
  - Sonar Distance Sensors
  - Controllable via Smart Phone App