Artificial Intelligence

Robotics

Tim Niemueller and Sumedha Widyadharma

Term and History

- Term comes from Karel Capek's play "R.U.R. Rossum's universal robots"
- Robots comes from the Czech word for corvee
- Manipulators first start in development of robots, they were machines that humans controlled remotely
- Numeric control first used in 1952 at MIT, allows very precise movement in relation to a given coordinate system
- These techniques lead to first programmable manipulator which was installed in 1961
- Mobile robots invented for automated transportation and driver-less cars
- Mostly used for "Get and Bring" services
- First humanoid robot in 1975 (Japan)

Definition: What is a Robot?

- Robots are physical agents
- They manipulate the physical world
- Equipped with sensors to perceive environment
- Effectors to assert physical forces on environment
- Three main categories
  - Manipulators
  - Mobile robots
  - Humanoid robots

Topics

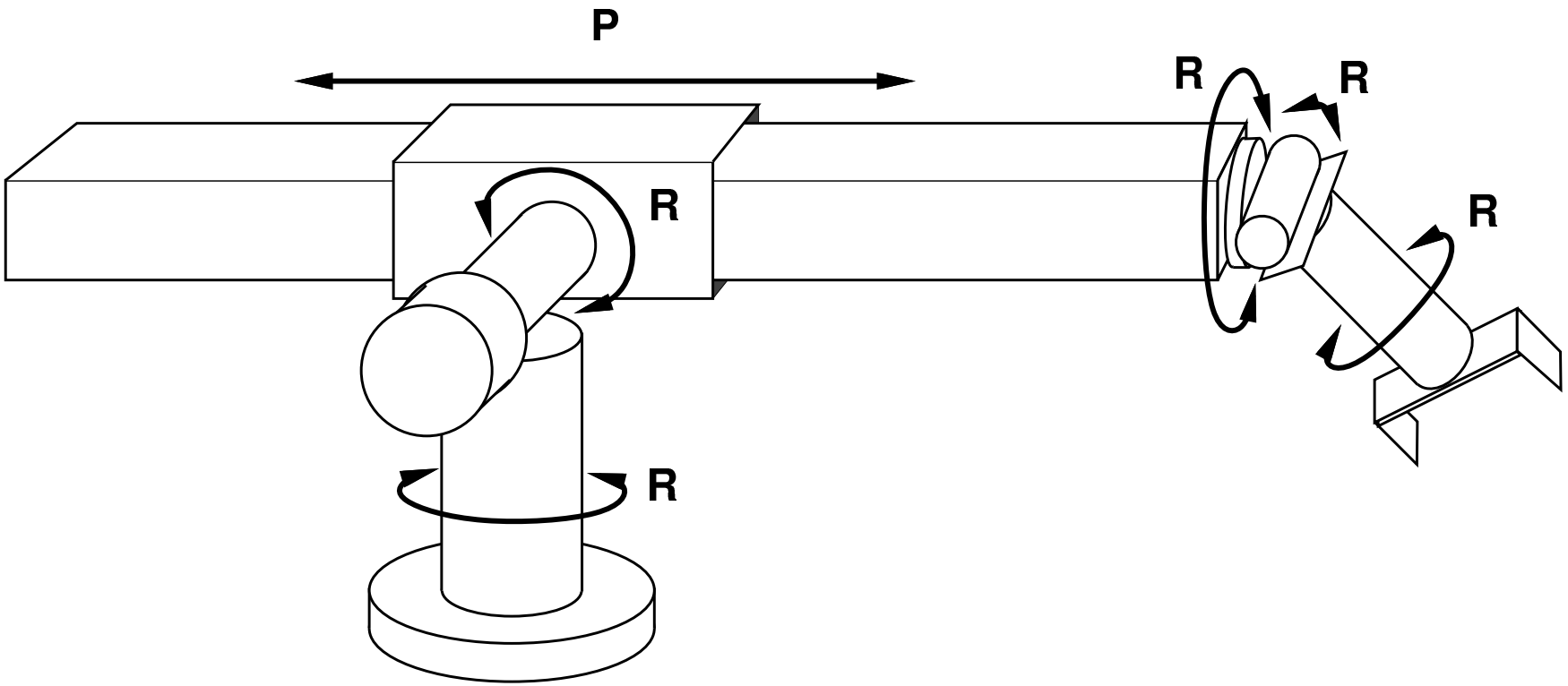
- Robot Hardware
- Localization
- Path Planning and Moving
- Reality

Robot Hardware – Sensors

- Used to perceive the robot's environment
- Two types of sensors
  - *Passive sensors*: Camera, . . .
  - *Active sensors*: Sonar, Radar, Laser scanners, . . .
- Three classes of sensors:
  - *Range finders* – Used to detect distance to nearby objects.
  - *Imaging sensors* – These are cameras that provide images of the environment. Image sensors are becoming more and more common. Used for localization and object tracking.
  - *Proprioceptive sensors* – These sensors tell the robot about its own state (Example: shaft decoder in revolute joints).

Robotic Hardware – Effectors

- *Effectors* are the means by which robots manipulate their environment, move and change the shape of their body
- Abstract concept of *degrees of freedom (DOF)* to understand effectors
  - → We count one degree of freedom for each independent direction in which a robot or one of its effectors can move
  - → Example ”Autonomous Underwater Vehicle (AUV)”: It has six degrees of freedom: Three for its (x, y, z) location in space and three for its orientation (also known as yaw, roll and pitch)
  - → Six DOFs are needed to place an object at a particular point in a particular orientation
  - → Manipulator has exactly six DOFs: Five revolute joints (R) and one prismatic joint (P).
  - → In general robots with more DOF than needed are easier to control and program



The Stanford Manipulator

Robotic Hardware – Effectors

Mobile robots are special

- Number of DOF do not need to have corresponding actuated elements
- Example ”car in a plane”: You can move the car forward and backward and turn, giving it two DOFs. But the car’s kinematic confi guration is three-dimensional: you can maneuver to any (x, y) point, in any orientation. So it has three *effective* DOFs but only two *controllable* DOFs
- We say a robot is *nonholonomic* if it has more effective DOFs than controllable DOFs and *holonomic* otherwise
- Holonomic robots are easier to control (think of parking the car: it would be much easier to move the car sideways)

Robotic Hardware – Movement

- Mobile robots have special effectors for locomotion
- Common types are wheels, tracks and legs
- *Differential* drive:
  - → Two independently actuated wheels – one on each side of the robot
  - → When moving at the same speed the robot moves straight
  - → When moving in opposite directions the robot turns on the spot
- *Synchro* drive:
  - → Each wheel can move and turn around its own axis
  - → This can lead to chaos
  - → Programmer (or engineer) needs to assure the constraint that all wheels always point in the same direction and move at the same speed

Robotic Perception

- Perception is the process of mapping raw sensor data into an internal representation
- A good representation should meet three criteria: It should
  - → contain enough information for the robot to make a right decision
  - → be structured in a way that it can be updated effi ciently
  - → be natural, meaning that internal variables correspond to natural state variables in the physical world

Localization – Basics

- Very generic perception task – and one of the most pervasive ones
- Problem of determining where things are
- Knowing where things are is a requirement for the robot for any successful interaction with the physical world
  - → Manipulators need to know where the objects are that they have to deal with
  - → Mobile robots must know where they are to fi nd their way to the target location
- There are three increasingly diffi cult flavors of localization problems:
  - → *Tracking* – Initial state of the object to be localized is known ⇒ track this object.
  - → *Global localization* – Initial location of the object is unknown ⇒ robot has to fi nd the object. When found this becomes a tracking problem.
  - → *Kidnapping* – Take the object and place it somewhere else ⇒ Robot has to localize it without knowing anything

Pose Description and simple Geometry

- To keep things simple we assume:
  - → Robot moves slowly in a plane
  - → Robot has an exact map of the environment.
  - → No Gaussian noise (of course this is not true in a real environment)
- Describe pose of robot by its cartesian coordinates (x, y) and its head θ.
  Formalize as vector $X_t = (x_t, y_t, \theta_t)^\top$
- Defi ne transition with $t \mapsto t + 1$ with angular velocity ω_t and translational velocity v_t for small time intervals Δt.
- Robots pose is $X_t = (x_t, y_t, \theta_t)^\top$. Landmark at location $(x_i, y_i)^\top$.

Transition and Sensor Model:

- **Deterministic transition model:**

$$\hat{X}_{t+1} = f(X_t, \underbrace{v_t, \omega_t}_{a_t}) = X_t + \begin{pmatrix} v_t \Delta t \cos\theta_t \\ v_t \Delta t \sin\theta_t \\ \omega_t \Delta t \end{pmatrix}$$

- **Sensor model:**

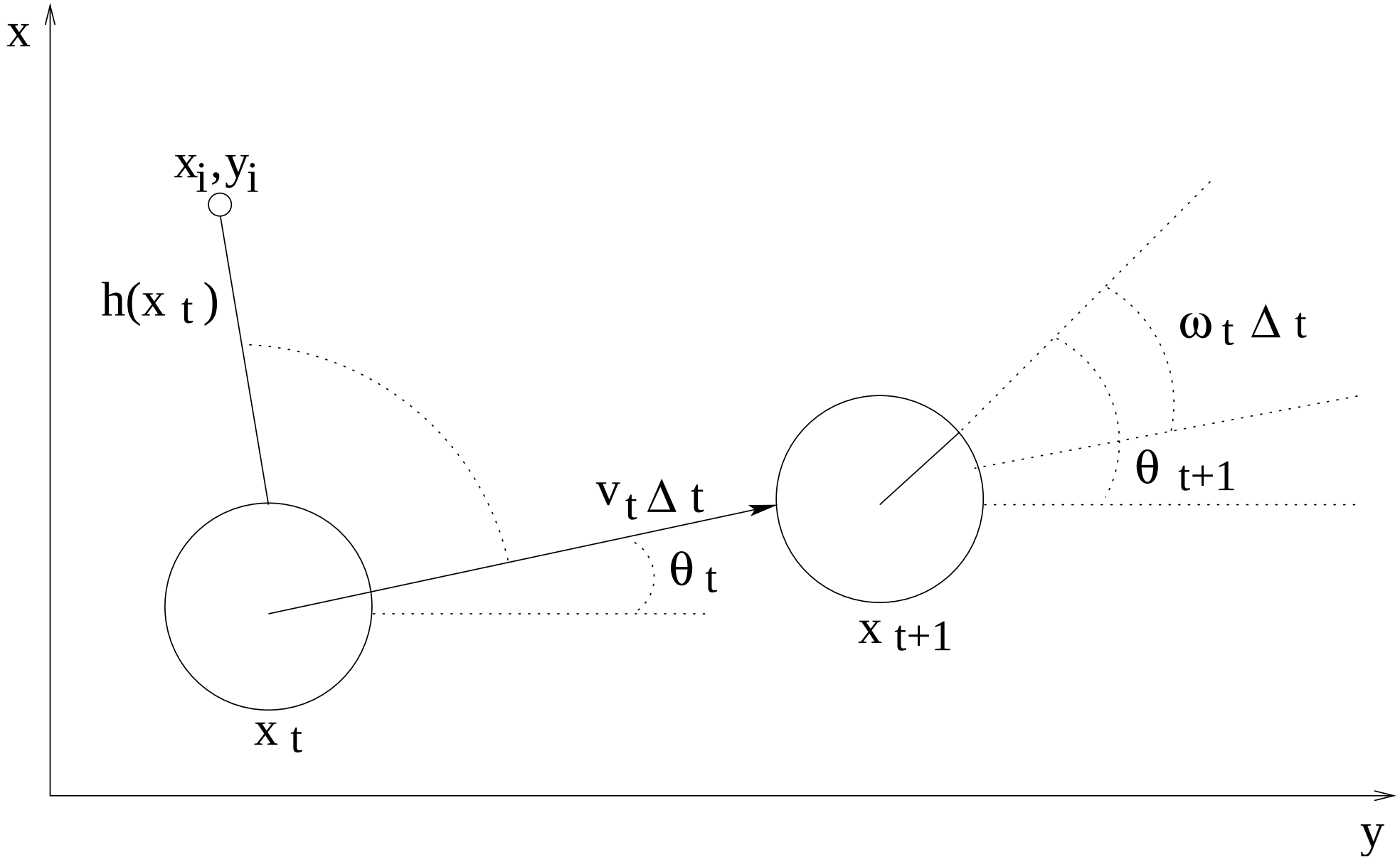
  → Sensors detect stable and recognizable features called *landmarks*.

- Range and bearing can be calculated by

$$\hat{z}_t = h(x_t) = \begin{pmatrix} \sqrt{(x_t - x_i)^2 + (y_t - y_i)^2} \\ \arctan\frac{y_i - y_t}{x_i - x_t} - \theta_t \end{pmatrix}$$

- Alternative: *range scan model*. Sensors produce a vector of range values $z_t = (z_1, \ldots, z_M)^\top$.

- Range scan model more indenpendent – landmark model can provide immediate localization


x
x_i, y_i
$h(x_t)$
$\omega_t \Delta t$
θ_{t+1}
$v_t \Delta t$
θ_t
x_{t+1}
x_t
y

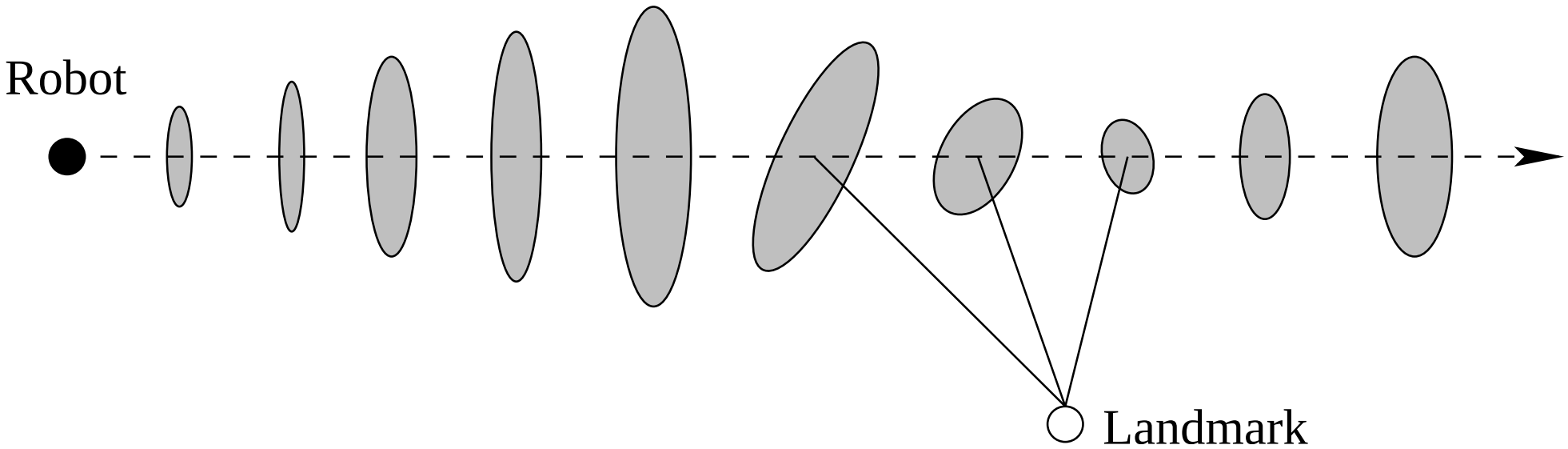
Monte Carlo Localization (MCL)

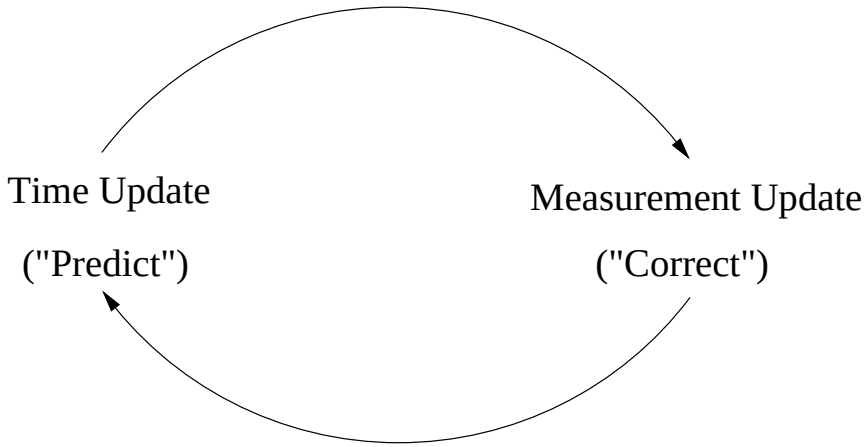
- MCL is a particle filtering algorithm. Requirements:
  - Map of the environment (a priori known or learned)
  - Appropriate sensor model and motion model
- Create population of N samples by a given probabilistic distribution. Samples shown as particles in the figure. Then a continuous update cycle is started with the following steps:
  - Each sample is propagated forward by sampling the next state value X_{t+1} given the current value X_t for the sample, and using the transition model given: $P(X_{t+1} \mid x_t)$.
  - Each sample is weighted by the likelihood it assigns to the new evidence: $P(e_{t+1} \mid x_{t+1})$
  - Resample a new population of N samples. New samples are selected from current population; probability that a particular sample is selected is proportional to its weight. New samples are unweighted.
- As the robot gathers more knowledge the particles concentrate at one or more points.
- At some point in time all points are in one cluster point. That is the location of the robot in the field.


Robot position
Robot position
Robot position

Extended Kalman Filter (EKF)

- Effi cient computational (recursive) solution to the least-squares method
- It is very powerful in many aspects: it supports estimations of past, present and future (with the latter one being the important thing for robotics)
- Continuous update cycle with alternating prediction and correction phases
- The robots state X_t is calculated by its posterior probability with $P(X_t \mid z_{1:t}, a_{1:t-1})$ by a Gaussian
- Problem: Gaussian only closed under linear modules. Robotic motion and sensor models are nonlinear $\Rightarrow$ models must be linearized
- This is done by a local approximation of a nonlinear function by a linear function.
- When using (fi rst degree) Taylor expansion for approximation this is called Extended Kalman Filter
- This is used to get an estimated location based on previous estimate and current sensor data (shown as ellipse in fi gure on next page).
- When spotting a landmark the uncertainty can be reduced while moving without such causes growing uncertainty


Robot
Landmark


Time Update
("Predict")
Measurement Update
("Correct")

Planning to move

There are different kinds of movement problems:

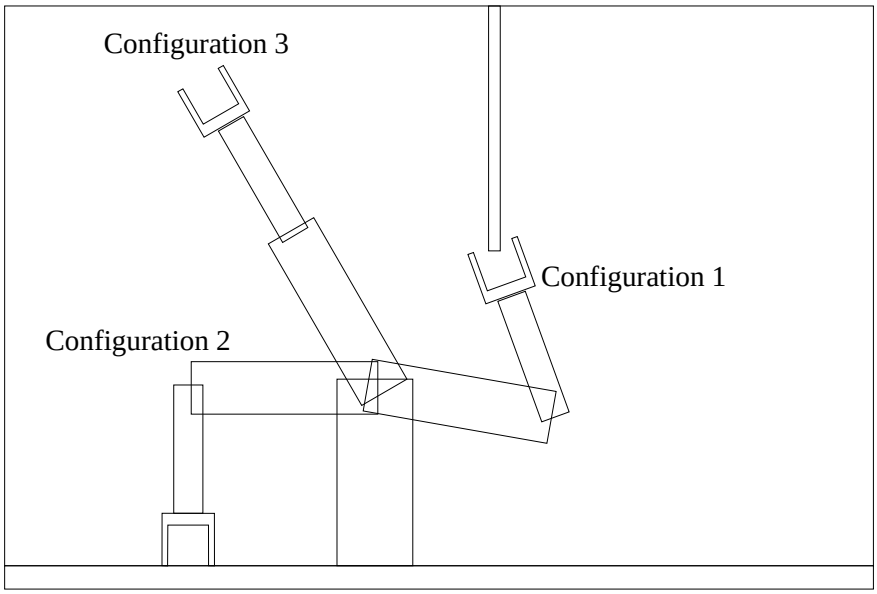
- The *point-to-point* motion problem
  - → The main issue is to determine the right motions of a robots effectors to reach a certain target location.
  - → *Point-to-point* motion problems are the easier ones to solve.

- The *compliant* motion problem
  - → *Compliant motion* describes a robots motion when in contact with an obstacle.
  - → These problems have more constraints and are therefore more diffi cult to solve.

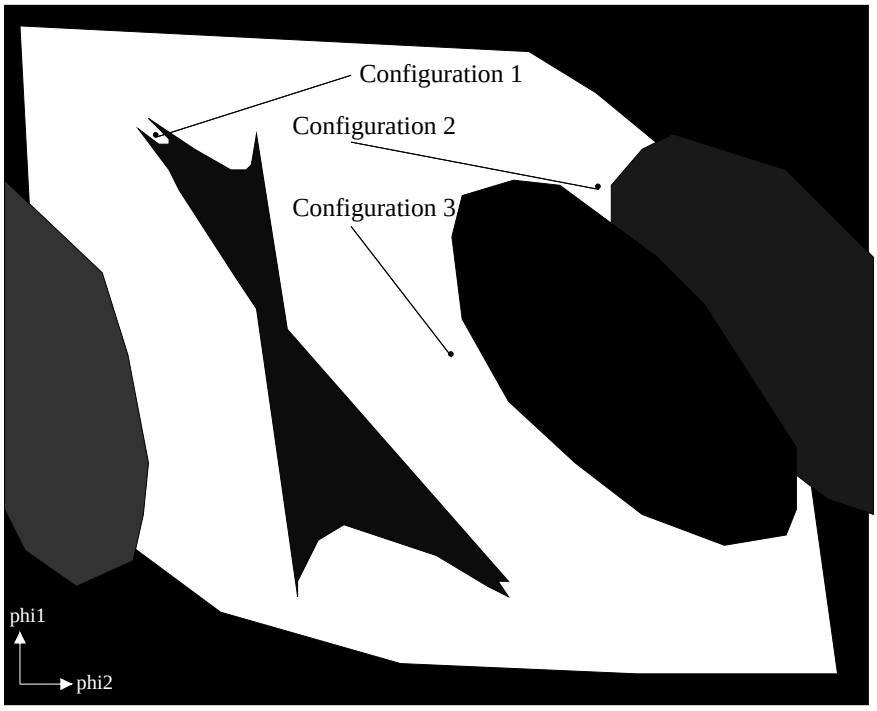
Workspace and Configuration Space

- Workspace model
  - → Robot and environment share the same coordinate system
  - → The state of the robot is represented by the carthesian coordinates of all its parts.
  - → Linkage constraints render path fi nding extremely diffi cult
- Confi guration space model
  - → Workspace coordinates need to be transformed
  - → The state of the robot is represented by the confi guration of its joints
  - → Constraints are easier to deal with in confi guration space

Configuration Space



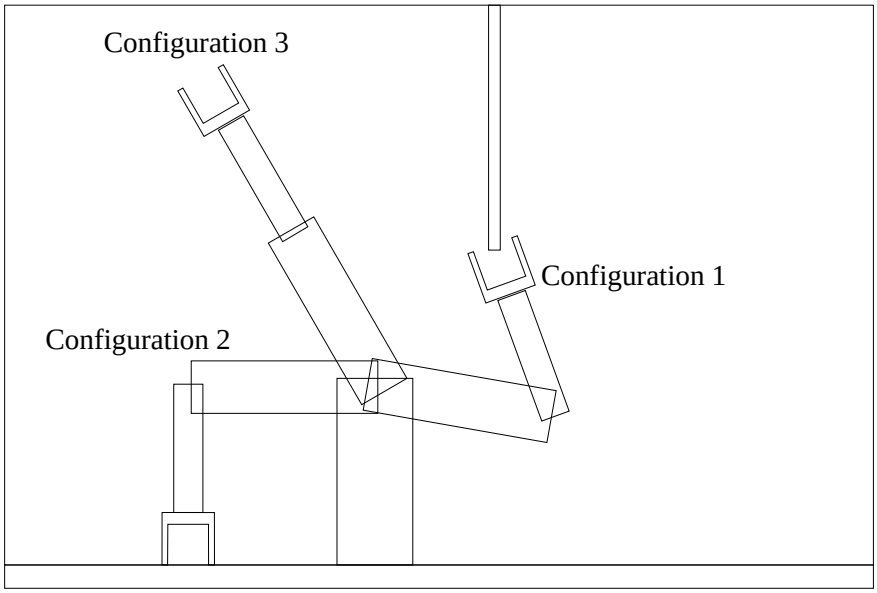
(a) The workspace



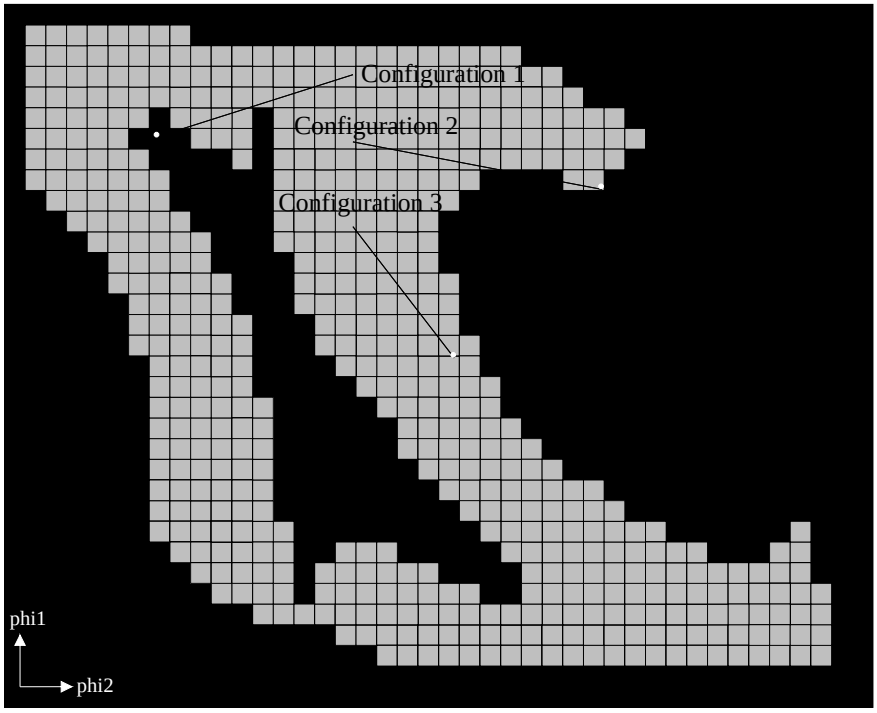
(b) The configuration space

A Way through free Space

Cell decomposition method



(c) The workspace



(d) Decomposed

Cell decomposition

- The free space is decomposed into (not necessarily square) cells
- Within the cell path fi nding is easy, which reduces path fi nding to a discrete graph search.

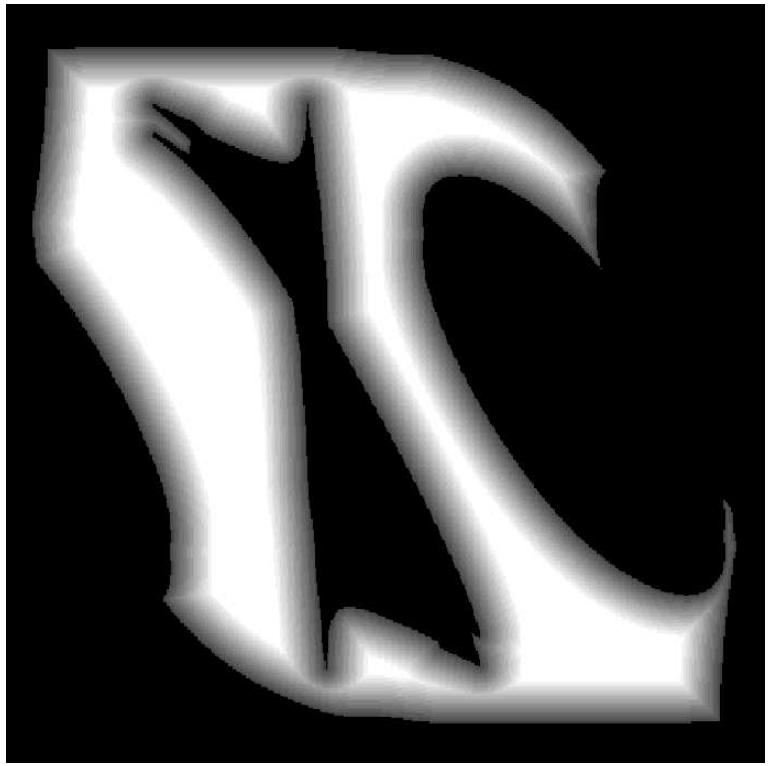
The cell decomposition method has the advantage of easy implementation, but suffers some defi - ciencies.

- *Polluted* cells require extra attention
  - → One solution would be to further subdivide these cells.
    This again leads to fast increase in complexity.
  - → Another could be to allow free shapes for cells.
- Scales poorly with the dimension of the confi guration space
- Resulting paths can have arbitrary sharp curves which can be impossible to perform in reality.
- Paths can come arbitrarily close to an obstacle.

Potential Fields

Potential fi elds are a way to solve several of the problems that the cell decomposition has.

- A potential fi eld is a function over the confi guration space.
- Its value grows with the proximity to an obstacle

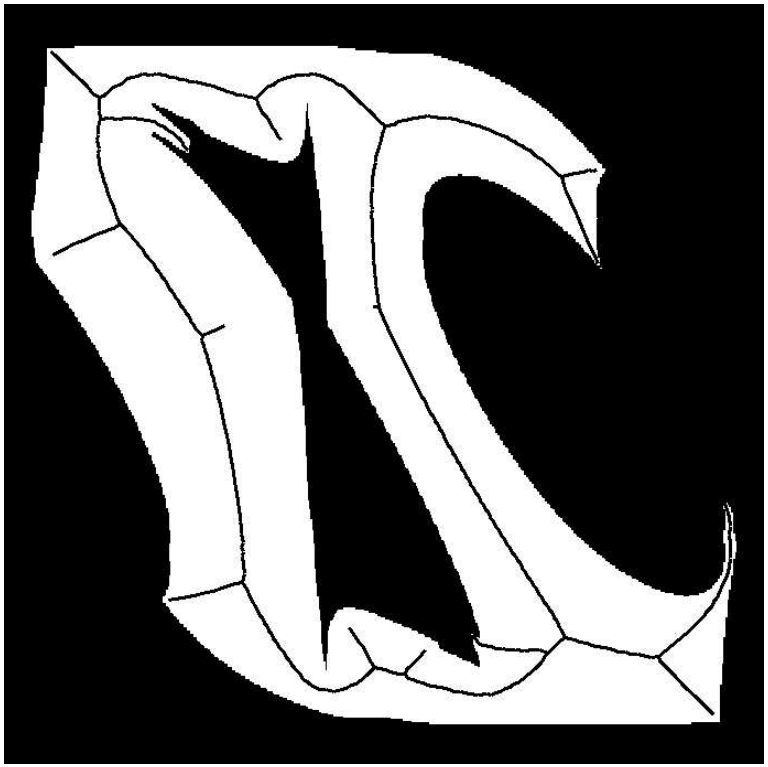


Skeletonization Methods

The idea of skeletonization lies in reducing the complexity of free space to a one-dimensional representation.

- *Voronoi diagrams*

  In this case the skeleton is a voronoi diagram of the free space.



Path planning with a Voronoi Diagram

A robot would use the following procedure to reach a point in confi guration space:

- Change its confi guration to a point on the skeleton

  Due to the nature of the skeleton this is always possible with a straight line through free space.

- Follow the skeleton to the point closest to the destination

- Take another straight line until the destination confi guration is reached

This method again reduces the path planning problem to a discrete graph seach along the voronoi diagram.

The found paths are in general not the shortest, but they have the advantage to maximise clearance.

Disadvantages of Voronoi Diagrams

There are several drawbacks when utilizing this method for path planning:

- This method is diffi cult to apply to higher dimensional confi guration spaces
- If the free space is wide open it tends to induce large detours
- As obstacles can take strange shapes in confi guration space voronoi diagrams can be quite hard to compute.

To avoid large detours one simply needs more routes.

- *Probabilistic roadmaps* introduce a completely different approach to path planning
  - → Pick a random confi guration
  - → If it is in occupied space discard it
  - → Repeat until N valid confi gurations have been found
  - → Connect any two points which have an ’easy’ connection (e.g. a straight line)
  - → add the start and end point to the resulting graph

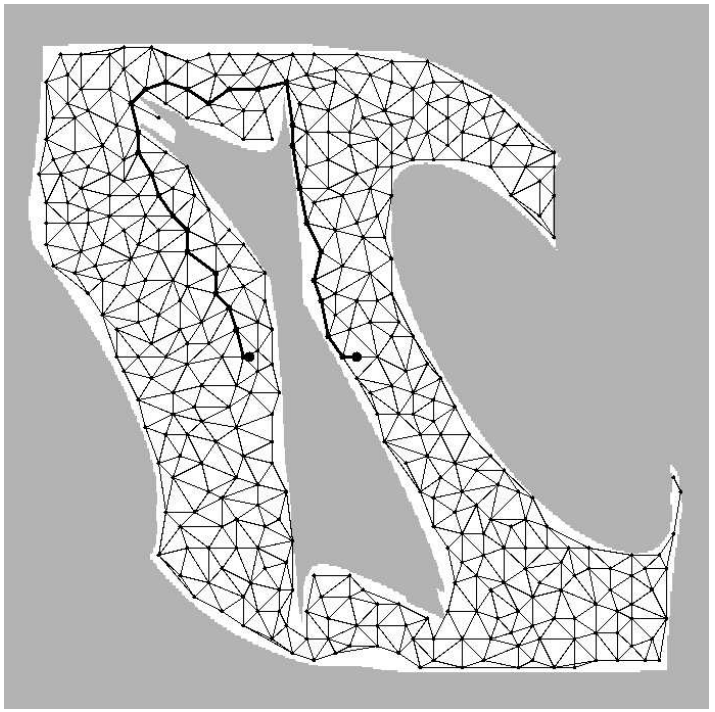
More on Probabilistic Roadmaps

Probabilistic roadmaps are theoretically incomplete.

Possible improvements:

- To introduce more points
- To concentrate points in areas that are likely to contain a path
- To take geometric properties of the confi guration space into account

With these improvements, the method normally scales better to high-dimensional confi guration spaces than most other path planning techniques.



Moving a robot

- Forces are exerted on effectors to move them
- Robots can only follow arbitratry paths at arbitrary slow speeds
- Dynamic state models
  - → *Dynamic state models* would generate better motion plans
  - → They are unusable for anything other than trivial robots
- *Controllers*
  - → Proportional integrative derivative controllers (PID controller)

$$a_t = K_P(y(t) - x_t) + K_I \int (y(t) - x_t)dt + K_D \frac{\partial (y(t) - x_t)}{\partial t}$$

Potential Field Control and Reactive Control

- Potential fi eld control
  - → Target confi guration is an universal attractor
  - → Obstacles emit a repellent force
  - → The robot simply follows the resulting gradients
  - → No motion planning required
- Reactive control
  - → In cases where accurate models are not available
  - → Based on rulesets
  - → Reacts on immediate environment feedback
  - → No preplanning needed

Reality

Today robots are employed in many areas:

- Industry/Agriculture
  - → Melon harvesters
  - → Ore transporters
  - → Sewer mappers
  - → Paint strippers
- Transport
  - → Container transporters that load/unload cargo ships
  - → Medication and food transport systems in hospitals
  - → Autonomous helicopters to reach remote areas
- Hazardous environment
  - → They help clear nuclear waste (e.g. in Chernobyl)
  - → They searched the remains of the WTC for survivors
  - → They help destroy ammunition all over the world

A real-life Example: The AllemaniAC Robocup Team

- Sensors
  - → Laser Range Scanner 30 cm above the floor to detect obstacles (not the ball)
  - → Camera which can be panned horizontally and vertically
  - → Shaft decoders in drives
- Effectors
  - → 2 Motors with 2.5 kW each giving a speed of 12 km/h
  - → Shoot effector – electro-magnetic spools
- Two Pentium III (933 MHz) CPUs – One for image recognition other for higher level tasks
- Uses a form of Monte Carlo Localization
- Base station besides the fi eld as communication point