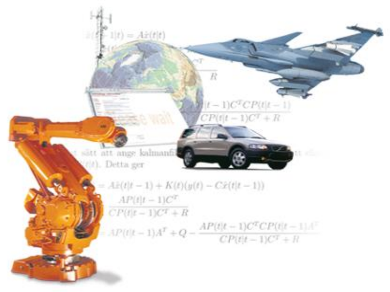
Path and trajectory planning



Lecture 5
Mikael Norrlöf

Up till now

- Lecture 1
  - Rigid body motion
  - Representation of rotation
  - Homogenous transformation
- Lecture 2
  - Kinematics
    - Position
    - Velocity via Jacobian
    - DH parameterization
- Lecture 3
  - Lagrange’s equation
  - (Newton Euler)
  - Parameter identification
    - Experiment design
    - Model structure
- Lecture 4
  - Robot Motion Control Overview
  - Current and Torque Control
  - Control Methods for Rigid and Flexible Robots
  - Interaction with the environment

Outline

- Path vs trajectory
- Standard path planning techniques in industrial robots
- Trajectory generation – an introduction to the problem
- More general path planning algorithms
  - Potential field approach
  - An introduction to Probabilistic Road Maps (PRMs)
- Lab session: *date/time to be decided*

Path and trajectory planning

What is the difference between path and trajectory?

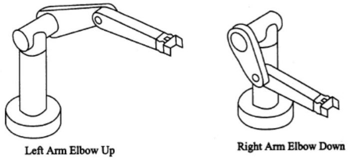
Path: Only geometric considerations. The way to go from config a to b.

Trajectory: Include time, i.e., consider the dynamics.

Compare: kinematics versus dynamics.

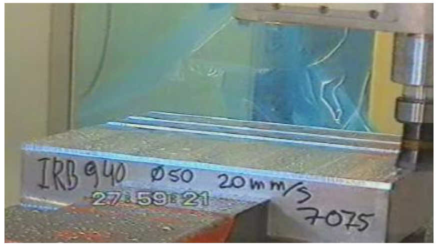
Path planning

- In industrial robot applications two path planning modes can be identified
  - Change configuration



Left Arm Elbow Up   Right Arm Elbow Down

  - Perform an operation



Path planning

- In industrial robot applications two path planning modes can be identified
  - Change configuration

    In RAPID (ABB's programming language)

    `MoveJ p10, v1000, z10, tool;`

  - Perform an operation

    `MoveL p20, v50, z1, tool;`

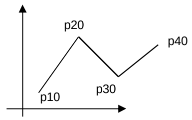
    `MoveC p30, p40, v25, fine, tool;`

Rapid, robot motion instructions

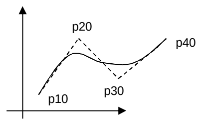
- Linear in
  - joint space: `MoveJ`
  - Cartesian space: `MoveL`
- Circle segment
  - `MoveC`

Via points

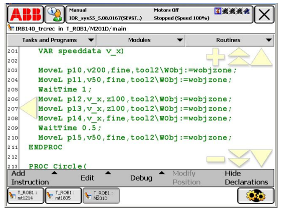
- Point-to-point (fine points in Rapid). Robot has to stop.



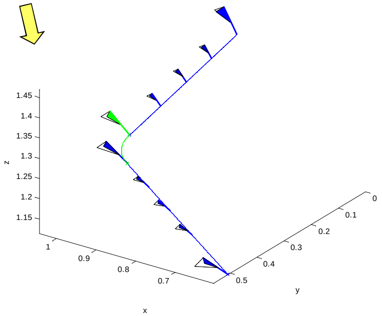
- Via points (zones in Rapid). Robot does not reach the programmed position.

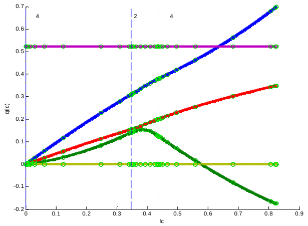


Geometric description



Inverse kinematics





One possible parameterization is *cubic splines*.

See MSc theses, “Path planning for industrial robots” by Maria Nyström and “Path generation in 6-DOF for industrial robots” by Daniel Forssman.

Chap 5 in “Modeling and Control of Robot Manipulators”, Sciavicco and Siciliano

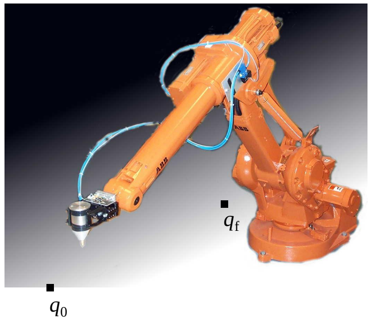
Orientation interpolation

- The orientation along the path is interpolated to get a smooth change of orientation.
- Given initial orientation (as a quaternion), q_0 and final orientation q_1 compute
  $$q_{01} = q_1 q_0^{-1}$$
  and interpolate the angle α from 0 to θ_{01} in
  $$q_{ip}(\alpha) = \langle \cos(\alpha/2),\ \sin(\alpha/2)\ s_{01} \rangle$$
  where (θ_{01}, s_{01}) is the angle-axis representation of q_{01}
- The interpolation scheme is often referred to as *slerp* interpolation

The trajectory generation problem

Optimal control!

$$\min t_f$$
$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u(t)$$
$$u(t) \in [\tau_{\min}, \tau_{\max}]$$
$$q(0) = q_0,\ \ q(t_f) = q_f,$$
$$\dot{q}(0) = 0,\ \ \dot{q}(t_f) = 0$$

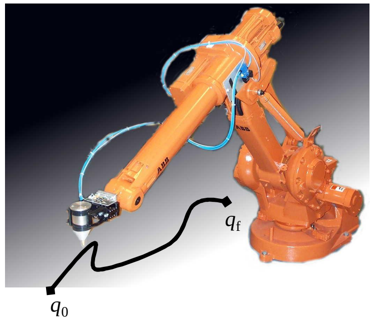


The trajectory generation problem

Optimal control!

$$\min t_f$$
$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u(t)$$
$$u(t) \in [\tau_{\min}, \tau_{\max}]$$
$$q(0) = q_0,\ \ q(t_f) = q_f,$$
$$\dot{q}(0) = 0,\ \ \dot{q}(t_f) = 0$$

The path is parameterized in some index $s \Rightarrow q_r(s)$ which introduces additional constraints.



The trajectory generation problem

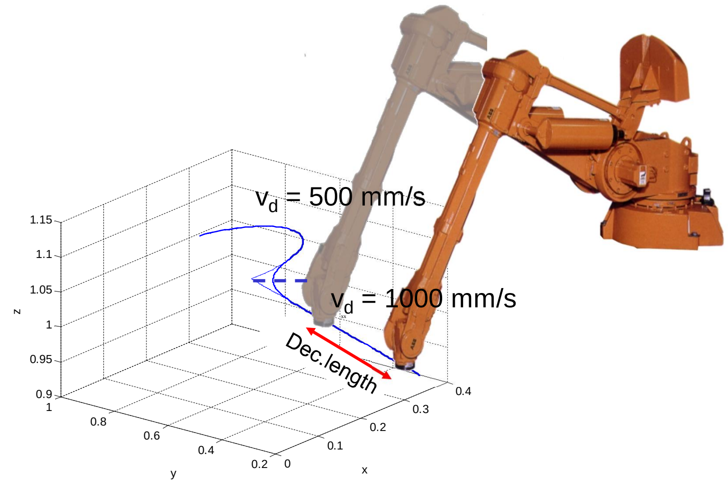
Resulting optimization problem

$$\min t_f$$
$$0 \leq v(t) \leq v_{\text{max}}$$
$$a_{\text{min}} \leq a(t) \leq a_{\text{max}}$$
$$|\dot{q}| \leq \dot{q}_{\text{max}}$$
$$\ddot{q}_{\text{min}} \leq \ddot{q} \leq \ddot{q}_{\text{max}}$$
$$\tau_{\text{min}} \leq M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \leq \tau_{\text{max}}$$
$$\ldots$$

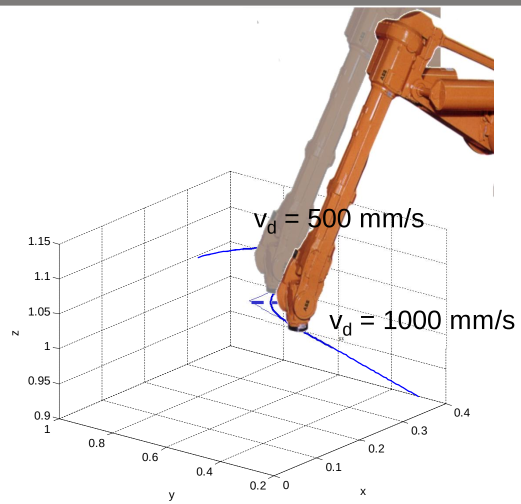
Complexity:

- 6 joints/actuators
- A robot can have > 100 constraints
- Multiple robots possible

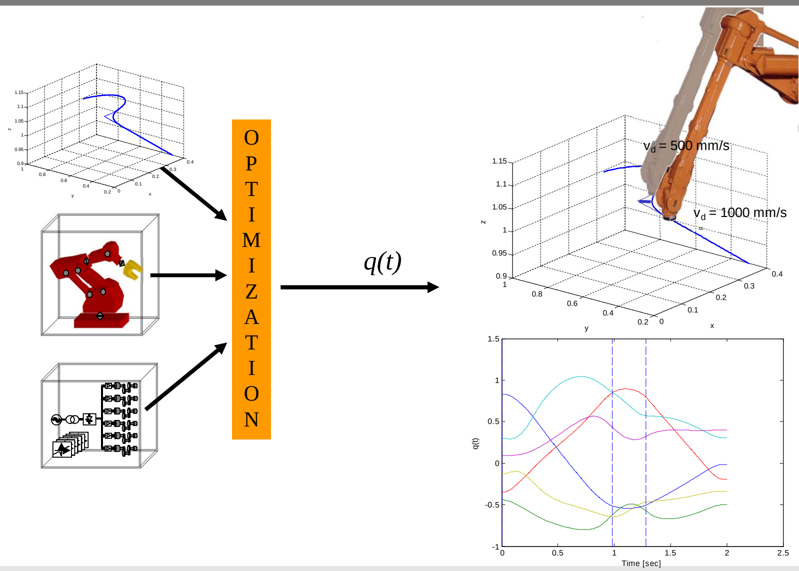
Trajectory generation problem



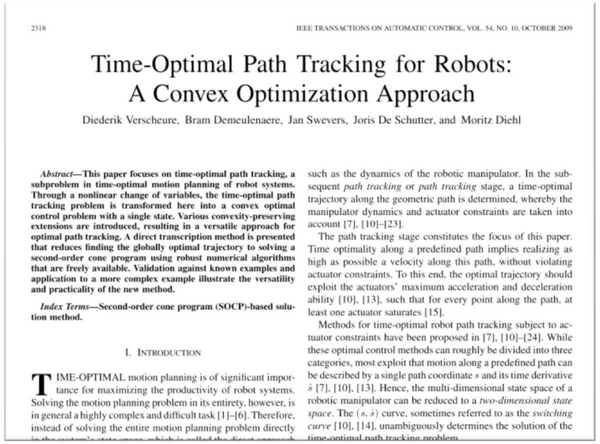
Trajectory generation problem



Dynamisk optimering



Solution



2318 IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 54, NO. 10, OCTOBER 2009

Time-Optimal Path Tracking for Robots: A Convex Optimization Approach

Diederik Verscheure, Bram Demeulenaere, Jan Swevers, Joris De Schutter, and Moritz Diehl

***Abstract*—This paper focuses on time-optimal path tracking, a subproblem in time-optimal motion planning of robot systems. Through a nonlinear change of variables, the time-optimal path tracking problem is transformed here into a convex optimal control problem with a single state. Various convexity-preserving extensions are introduced, resulting in a versatile approach for optimal path tracking. A direct transcription method is presented that reduces finding the globally optimal trajectory to solving a second-order cone program using robust numerical algorithms that are freely available. Validation against known examples and application to a more complex example illustrate the versatility and practicality of the new method.**

***Index Terms*—Second-order cone program (SOCP)-based solution method.**

I. Introduction

TIME-OPTIMAL motion planning is of significant importance for maximizing the productivity of robot systems. Solving the motion planning problem in its entirety, however, is in general a highly complex and difficult task [1]–[6]. Therefore, instead of solving the entire motion planning problem directly

such as the dynamics of the robotic manipulator. In the subsequent *path tracking* or *path tracking* stage, a time-optimal trajectory along the geometric path is determined, whereby the manipulator dynamics and actuator constraints are taken into account [7], [10]–[23].

The path tracking stage constitutes the focus of this paper. Time optimality along a predefined path implies realizing as high as possible a velocity along this path, without violating actuator constraints. To this end, the optimal trajectory should exploit the actuators' maximum acceleration and deceleration ability [10], [13], such that for every point along the path, at least one actuator saturates [15].

Methods for time-optimal robot path tracking subject to actuator constraints have been proposed in [7], [10]–[24]. While these optimal control methods can roughly be divided into three categories, most exploit that motion along a predefined path can be described by a single path coordinate s and its time derivative $\dot{s}$ [7], [10], [13]. Hence, the multi-dimensional state space of a robotic manipulator can be reduced to a *two-dimensional state space*. The $(s, \dot{s})$ curve, sometimes referred to as the *switching curve* [10], [14], unambiguously determines the solution of the

Dynamic scaling of trajectories

- The dynamic model can be rewritten in the form

$$D(q)\ddot{q} + \underbrace{C(q,\dot{q})\dot{q}}_{\Gamma(q)[\dot{q}\dot{q}]} + g(q) = \tau$$

- Let $r = ct$ (time scaling). The original speed/acc dep torque is

$$\bar{\tau}_s = D(q)\ddot{q} + \Gamma(q)[\dot{q}\dot{q}]$$

If time scaling is applied

$$\tau_s = \dot{r}^2\bar{\tau}_s + \ddot{r}D(q(r))q'(r)$$

and with linear time scaling

$$\tau_s = c^2\bar{\tau}_s$$

Application to a robot system

Use more than one robot to increase the flexibility in an application.

Here with application arc welding.

A more general path planning scheme

Configuration space

A complete specification of the location of every point on the robot is a *configuration* (Q).

The set of all configurations is called the *configuration space.*

q + kinematics give configuration.

Obstacles

The workspace of the robot is W. The subset of W occupied by the robot is $A(q)$.

The *configuration space obstacle* is defined as

$$QO = \{q \in Q \mid A(q) \cap O \neq \emptyset\}$$

with $O = \cup O_i$

The collision free configurations are

$$Q_{\text{free}} = Q \setminus QO$$

Collision detection

- A number of packages exists on the internet:
  - One example is from the group “team gamma” at Berkley
    http://gamma.cs.unc.edu/research/collision/packages.html
  - Another at University of Oxford
    http://web.comlab.ox.ac.uk/people/Stephen.Cameron/distances/
  - See also wikipedia
    http://en.wikipedia.org/wiki/Collision_detection
- An application where collision detection is used can be found in MSc thesis
  http://www.control.isy.liu.se/student/exjobb/xfiles/2050.pdf

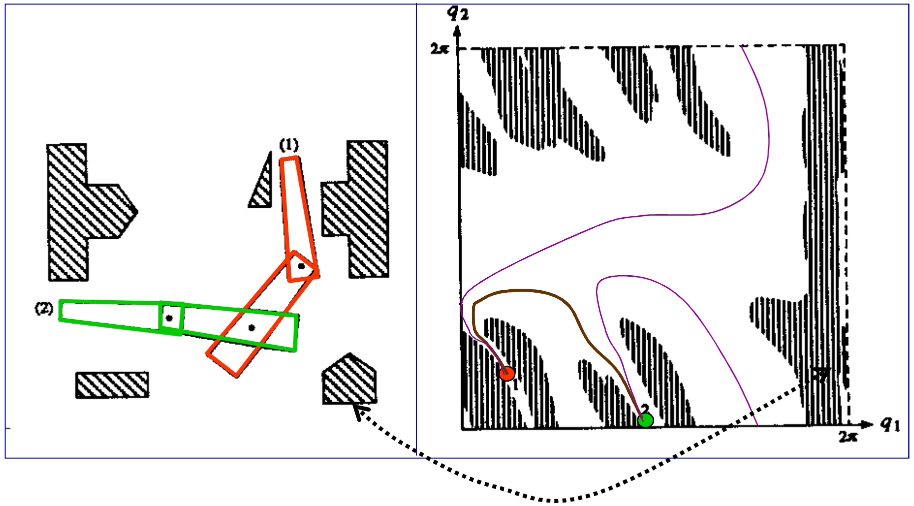
General formulation of the path planning problem

Find a collision free path from an initial configuration q_s to a final configuration q_f.

More formally $\gamma : [0,1] \to Q_{\text{free}}$
with $\gamma(0) = q_s$ and $\gamma(1) = q_f$.

Examples of methods:

- Path planning using potential fields
- Probabilistic road maps (PRMs)



Slide from: http://ai.stanford.edu/~latombe/

Artificial potential field

- Idea: Treat the robot as a point particle in the configuration space, influenced by an artificial potential field. Construct the field U such that the robot is attracted to the final configuration q_f while being repelled from the boundaries of QO.
- In general it is difficult/impossible to construct a field without having local minima.

Construction of the field *U*

- U can be constructed as an addition of an attractive field and a second component that repels the robot from the boundary of QO

$$U(q) = U_{att}(q) + U_{rep}(q)$$

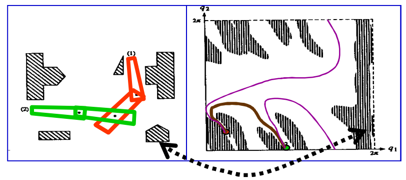
- Path planning can be treated as an optimization problem, finding the global minimum of $U(q)$. One simple approach is to use a gradient descent algorithm.

  Let $\tau(q) = -\nabla U(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q)$

Construction of the field *U*

Comments

- In general difficult to construct the potential field in configuration space (the field is often based on the norm of the min length to the obstacles)
- Easier to define the field in the robot workspace



- For an n-link manipulator a potential field is constructed for each DH-frame
- The link between workspace and configuration space is the Jacobian

The attractive field

- Conic well potential ($U_{att,i}(q) = \|o_i(q) - o_i(q_f)\|$)
- Parabolic well potential ($U_{att,i}(q) = \frac{1}{2} \xi_i \|o_i(q) - o_i(q_f)\|^2$)
- Parabolic well potential with upper bound

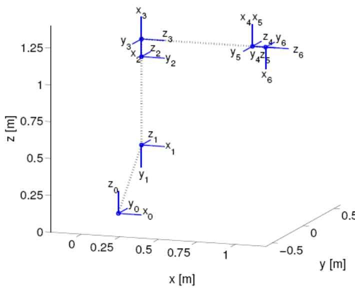
$$U_{att,i}(q) = \begin{cases} \frac{1}{2}\zeta_i \left\| o_i(q) - o_i(q_f) \right\|^2, & \left\| o_i(q) - o_i(q_f) \right\| \le d \\ d\zeta_i \left\| o_i(q) - o_i(q_f) \right\|^2 - \frac{1}{2} d^2 \zeta_i, & \left\| o_i(q) - o_i(q_f) \right\| > d \end{cases}$$

and

$$F_{att,i}(q) = \begin{cases} -\zeta_i \left(o_i(q) - o_i(q_f) \right), & \left\| o_i(q) - o_i(q_f) \right\| \le d \\ d\zeta_i \dfrac{o_i(q) - o_i(q_f)}{\left\| o_i(q) - o_i(q_f) \right\|}, & \left\| o_i(q) - o_i(q_f) \right\| > d \end{cases}$$

Path planning – obstacle free path

Notice:
Including only the origin of the DH-frames does not guarantee collision free path. (Additional points can however be added.)



The repulsive field

Criteria for the repulsive field to satisfy,

- Repel the robot from obstacles (never allow the robot to collide)
- Obstacles should not affect the robot when being far away

The repulsive field

One possible choice

$$U_{rep,i}(q) = \begin{cases} \frac{1}{2}\eta_i \left(\dfrac{1}{\rho(o_i(q))} - \dfrac{1}{\rho_0} \right)^2, & \rho(o_i(q)) \le \rho_0 \\ 0, & \rho(o_i(q)) > \rho_0 \end{cases}$$

with

$$F_{rep,i}(q) = \eta_i \left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(o_i(q))} \nabla \rho(o_i(q))$$

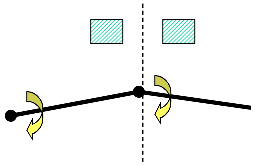
If the obstacle region is convex and b is the closest point to o_i

$$\rho(o_i(q)) = \left\| o_i(q) - b \right\|,$$

$$\nabla \rho(x) \Big|_{x = o_i(q)} = \frac{o_i(q) - b}{\left\| o_i(q) - b \right\|}$$

Comment, the “convex assumption”

- The force vector has a discontinuity
- Distance function not differentiable everywhere



Can be avoided if repulsive fields of distinct obstacles do not overlap.

Mapping the workspace forces into joint torques

If a force is exerted at the end-effector

$$J_v^T F = \tau$$

The Jacobian can be derived in all the points o_i.

Notice that the full Jacobian can be used when mapping forces and torques from workspace to torques in configuration space.

Path construction in configuration space

- Build the path using the resulting configuration space torques and an optimization algorithm

Gradient descent algorithm

1. $q^0 \leftarrow q_{\text{init}},\ i \leftarrow 0$
2. **IF** $q^i \neq q_{\text{final}}$
   $q^{i+1} \leftarrow q^i + \alpha^i \frac{F(q^i)}{||F(q^i)||}$
   $i \leftarrow i+1$
   ELSE return $< q^0, q^1 \cdots q^i >$
3. **GO TO** 2

Design parameters, α^i, ζ_i, η_i, ρ_0.

Typical problem: Can get stuck in local minima.

Escape local minima using randomization

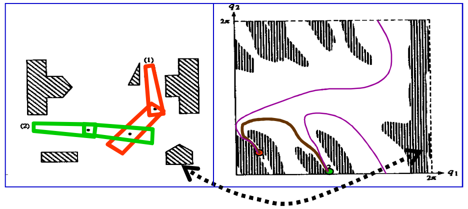
When stuck in a local minimum execute a random walk

1. $q^0 \leftarrow q_{\text{init}},\ i \leftarrow 0$
2. **IF** $q^i \neq q_{\text{final}}$
   $q^{i+1} \leftarrow q^i + \alpha^i \frac{F(q^i)}{||F(q^i)||}$
   $i \leftarrow i+1$
   ELSE return $< q^0, q^1 \cdots q^i >$
3. **IF** stuck in a local minimum
   execute a random walk, ending at q'
   $q^{i+1} \leftarrow q'$
4. **GO TO** 2

New problems:

- Detect when a local minimum is reached
- Define how the random walk should behave (how many steps, define the random terms, variance, distribution, …)

A more systematic way to build collision free paths

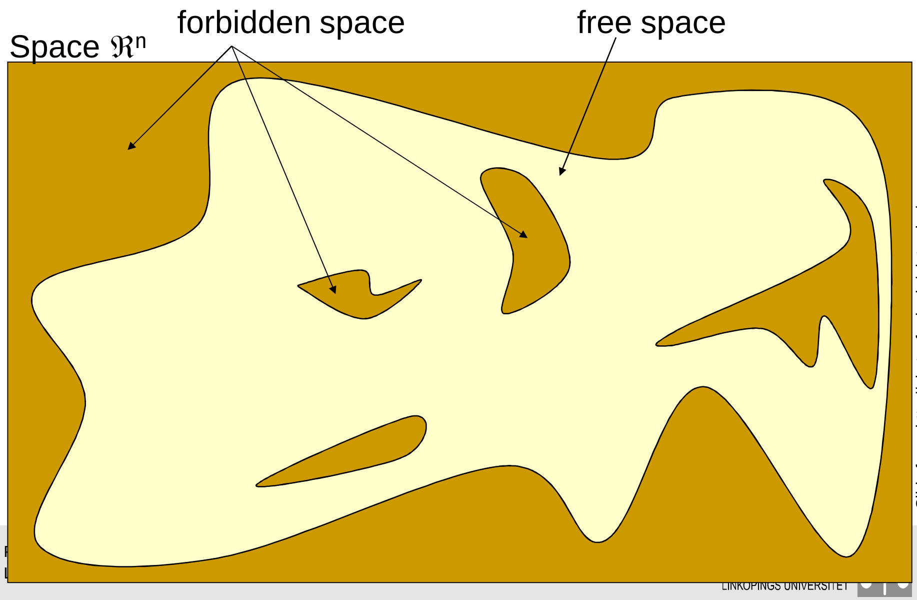


- The cost of computing an exact representation of the configuration space of a multi-joint articulated object is often prohibitive.
- But very fast algorithms exist that can check if an articulated object at a given configuration collides with obstacles.

→ Basic idea of Probabilistic Roadmaps (PRMs): Compute a very simplified representation of the free space by sampling configurations at random.

Slide from: http://ai.stanford.edu/~latombe/

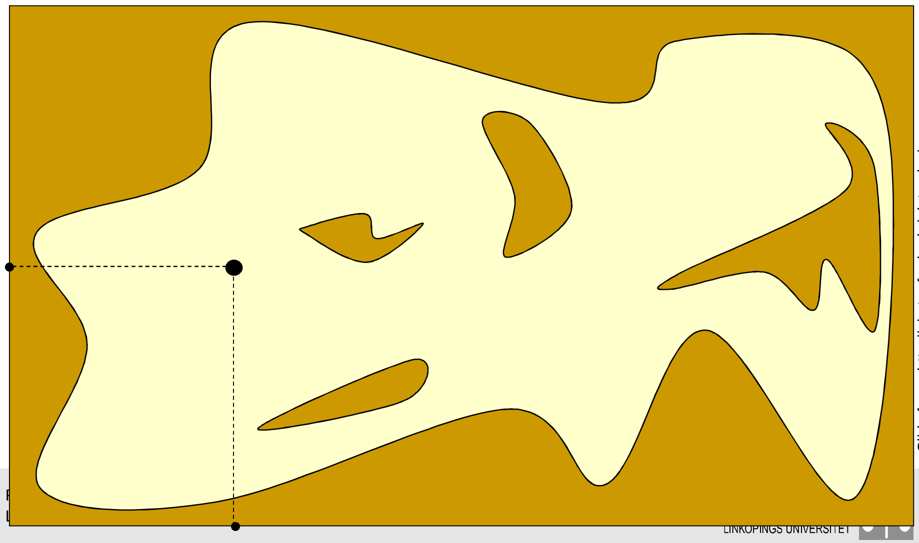
Probabilistic Roadmap (PRM)



Slide from: http://ai.stanford.edu/~latombe/

Probabilistic Roadmap (PRM)

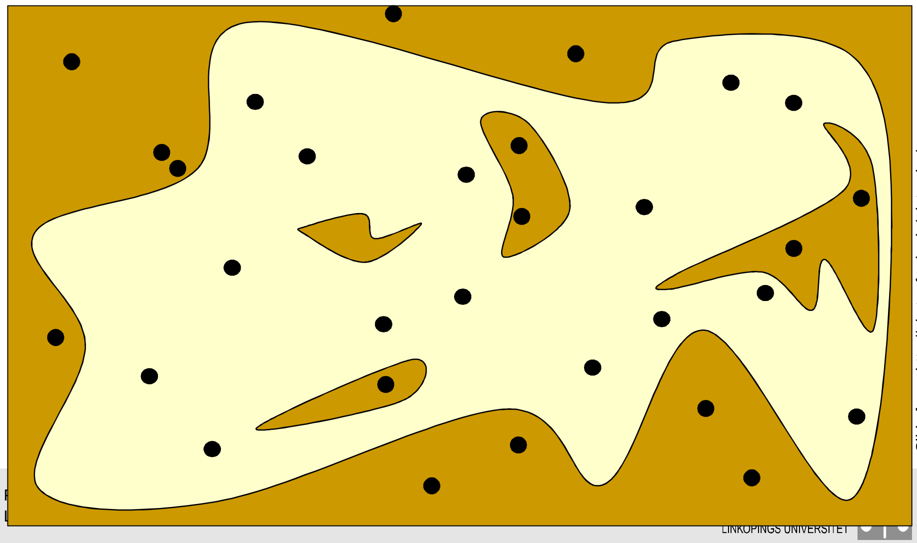
Configurations are sampled by picking coordinates at random



Slide from: http://ai.stanford.edu/~latombe/

Probabilistic Roadmap (PRM)

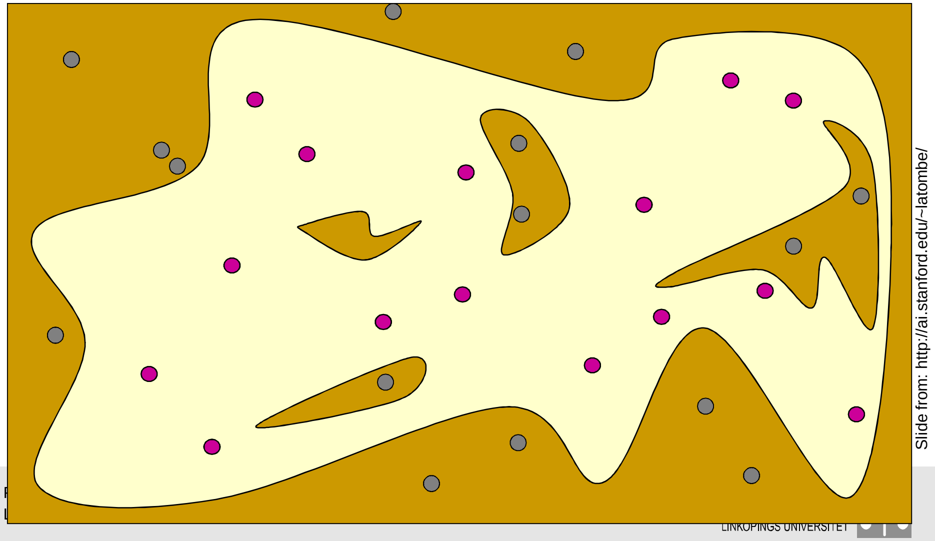
Configurations are sampled by picking coordinates at random



Slide from: http://ai.stanford.edu/~latombe/

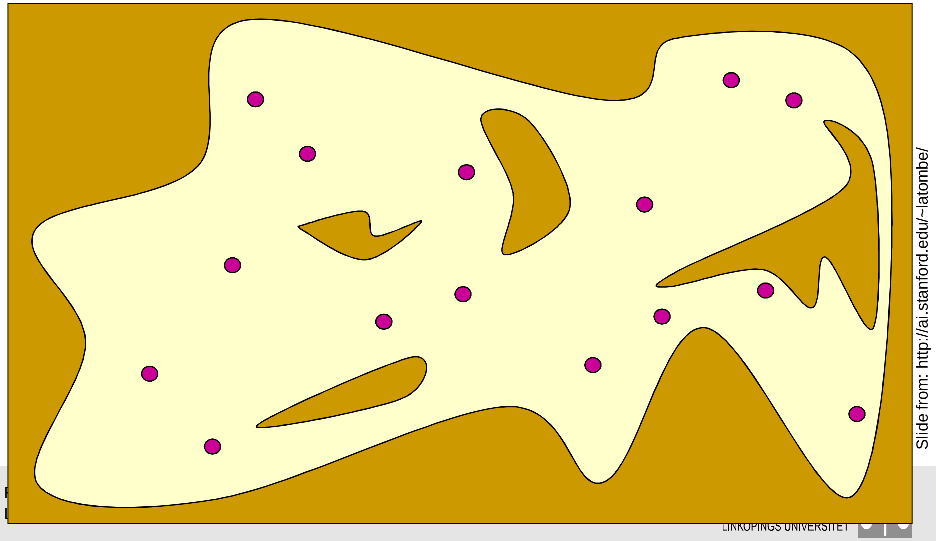
Probabilistic Roadmap (PRM)

Sampled configurations are tested for collision (in workspace!)



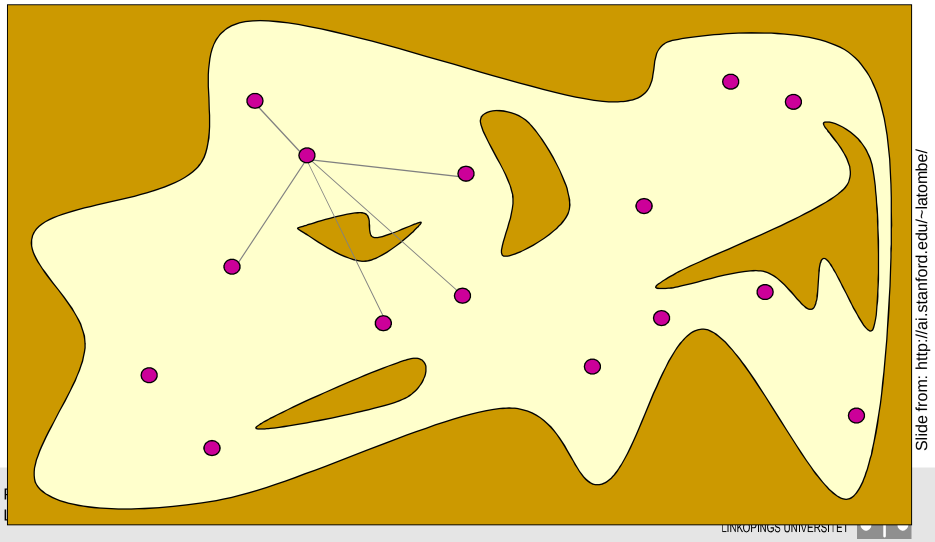
Probabilistic Roadmap (PRM)

The collision-free configurations are retained as “milestones”



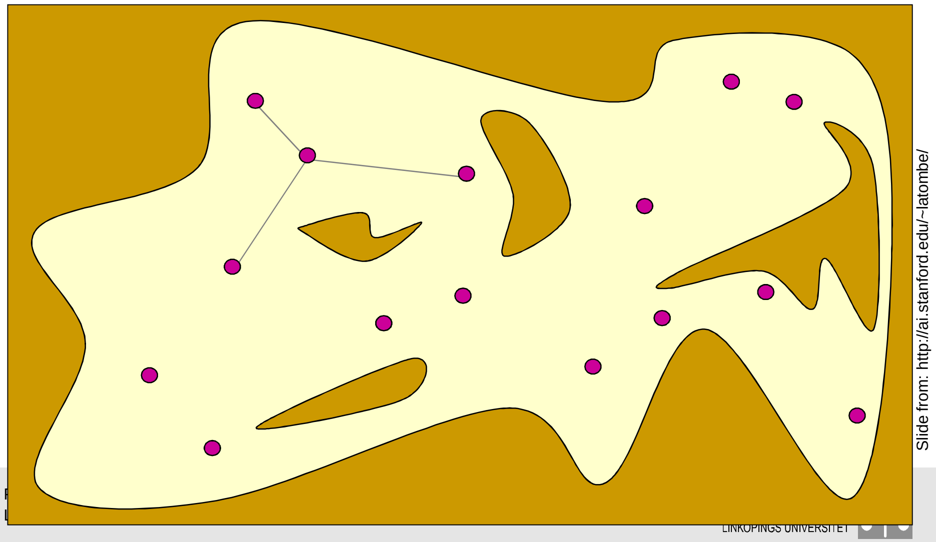
Probabilistic Roadmap (PRM)

Each milestone is linked by straight paths to its k-nearest neighbors



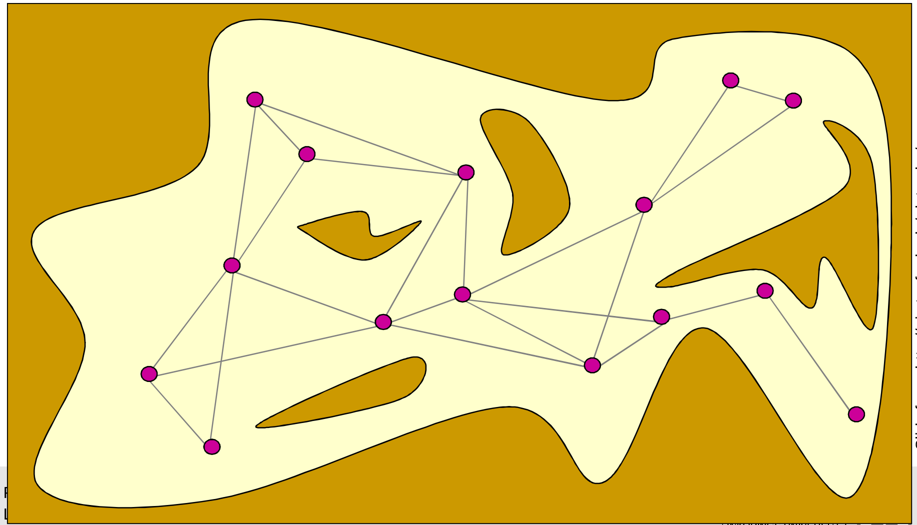
Probabilistic Roadmap (PRM)

Each milestone is linked by straight paths to its k-nearest neighbors



Probabilistic Roadmap (PRM)

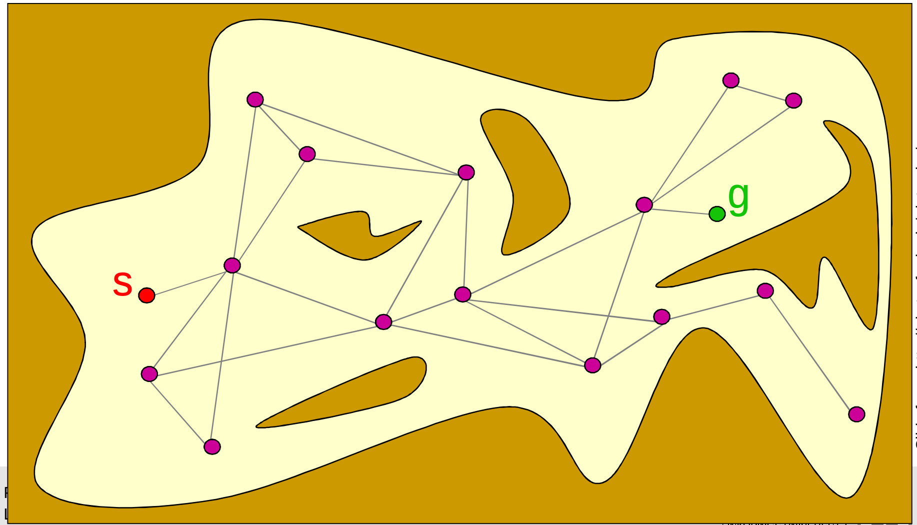
The collision-free links are retained to form the PRM



Slide from: http://ai.stanford.edu/~latombe/

Probabilistic Roadmap (PRM)

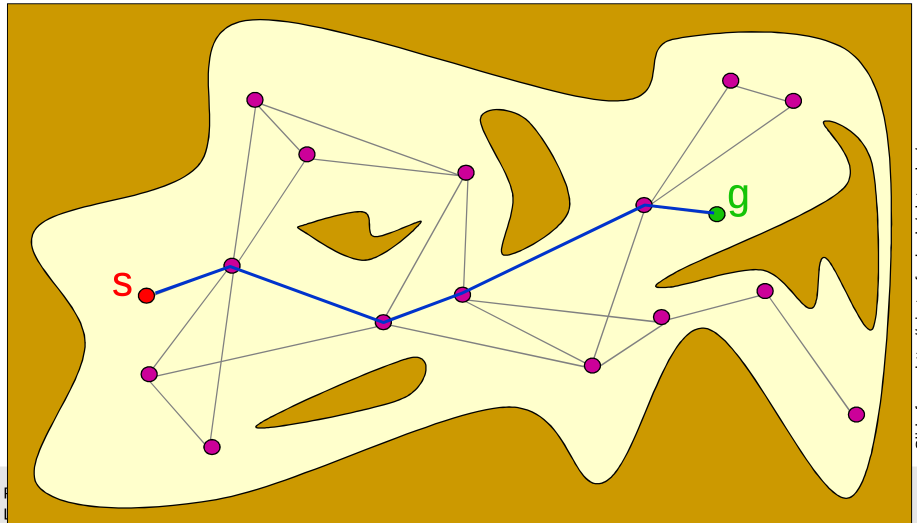
The start and goal configurations are included as milestones



Slide from: http://ai.stanford.edu/~latombe/

Probabilistic Roadmap (PRM)

The PRM is searched for a path from s to g



Slide from: http://ai.stanford.edu/~latombe/

Comments

- In industrial robotic applications the path planning problem is very much left to the user
- New ideas from mobile robotics (potential field algorithms, PRMs, etc. could be applied)
- Automatic planning algorithms highly complex

- The trajectory generation problem can be solved by applying optimal control techniques
- Conceptually easy to solve the offline problem
- Difficult to implement online

Comments

- Other planning requirements in many applications

Lab session

- Suggestion
  - Jan 31 – Time 9-15 (?)
  - Explore the possibilities in Rapid/RobotStudio
  - Exercise based on the previous lab but including multiple frames and also including moving frames (additional mechanical units)

Projects

- Kinematic redundancy
- Estimation and control
- Modeling and Identification
- Path planning and trajectory generation
  - Energy optimal
  - Sensor control (conveyor tracking)
  - …
- Daignosis

- ROS – Robot operating system
- Further develop the robot from the exercises – modeling and control
- Further explore the DH parameterization and the possibility to use it in RobotStudio
- …