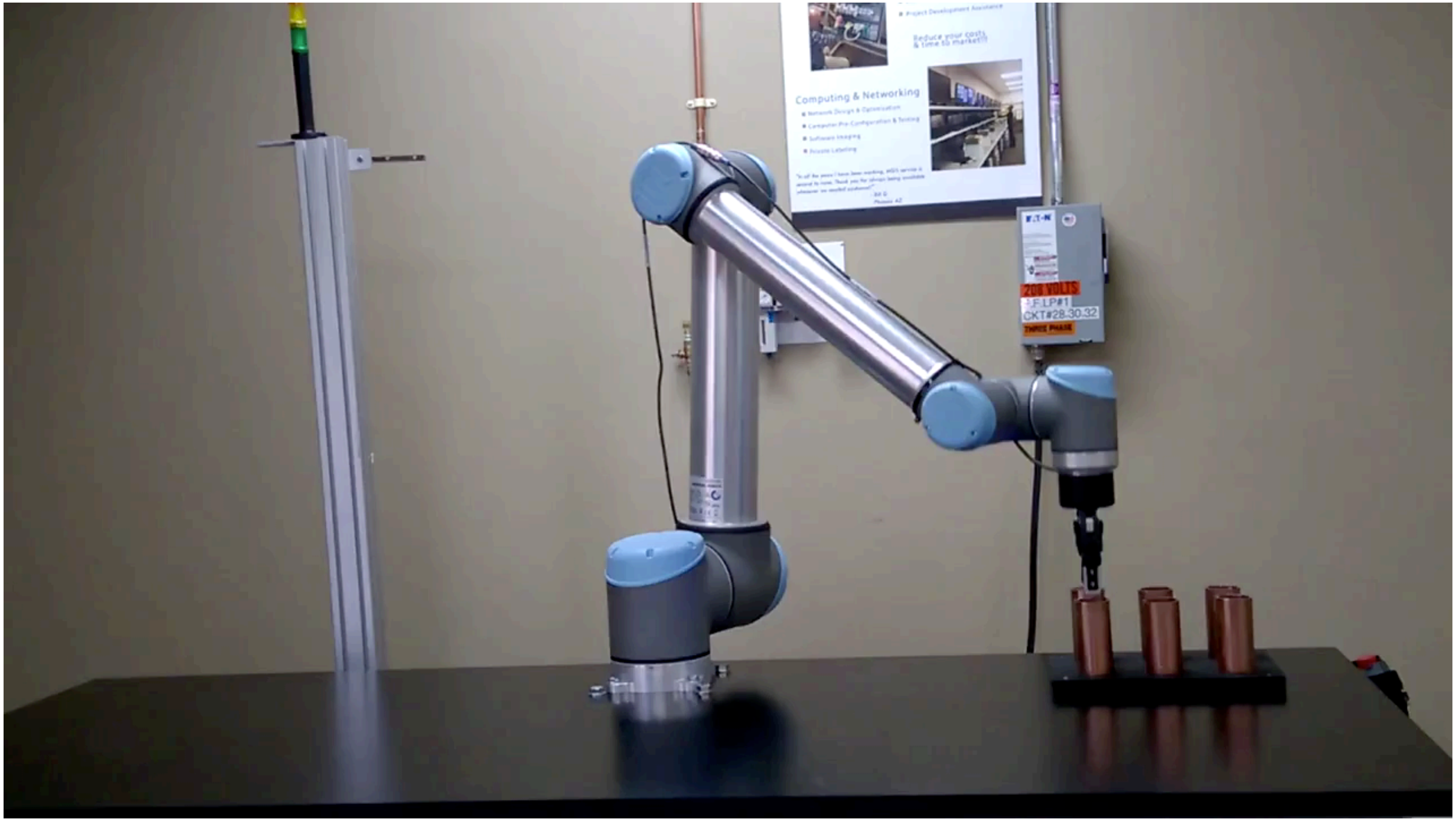
Robotics Review

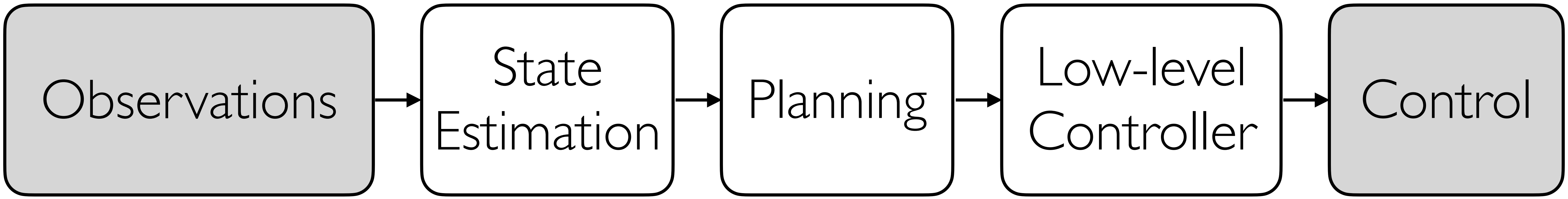
Saurabh Gupta

Robotic Tasks

Manipulation



Typical Robotics Pipeline

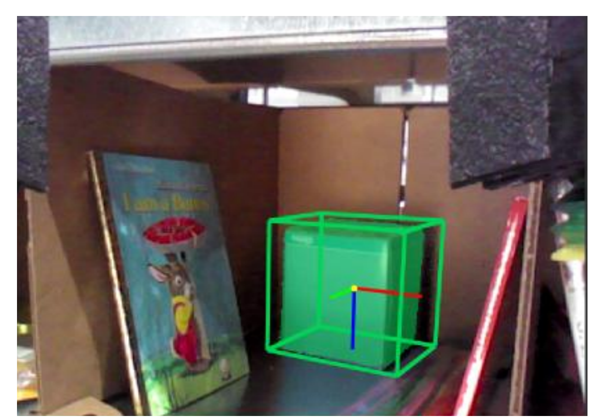


Typical Robotics Pipeline

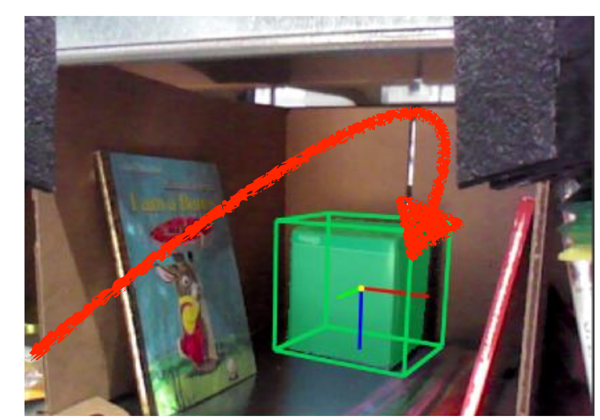
Observations → State Estimation → Planning → Low-level Controller → Control

Manipulation

Observed Images

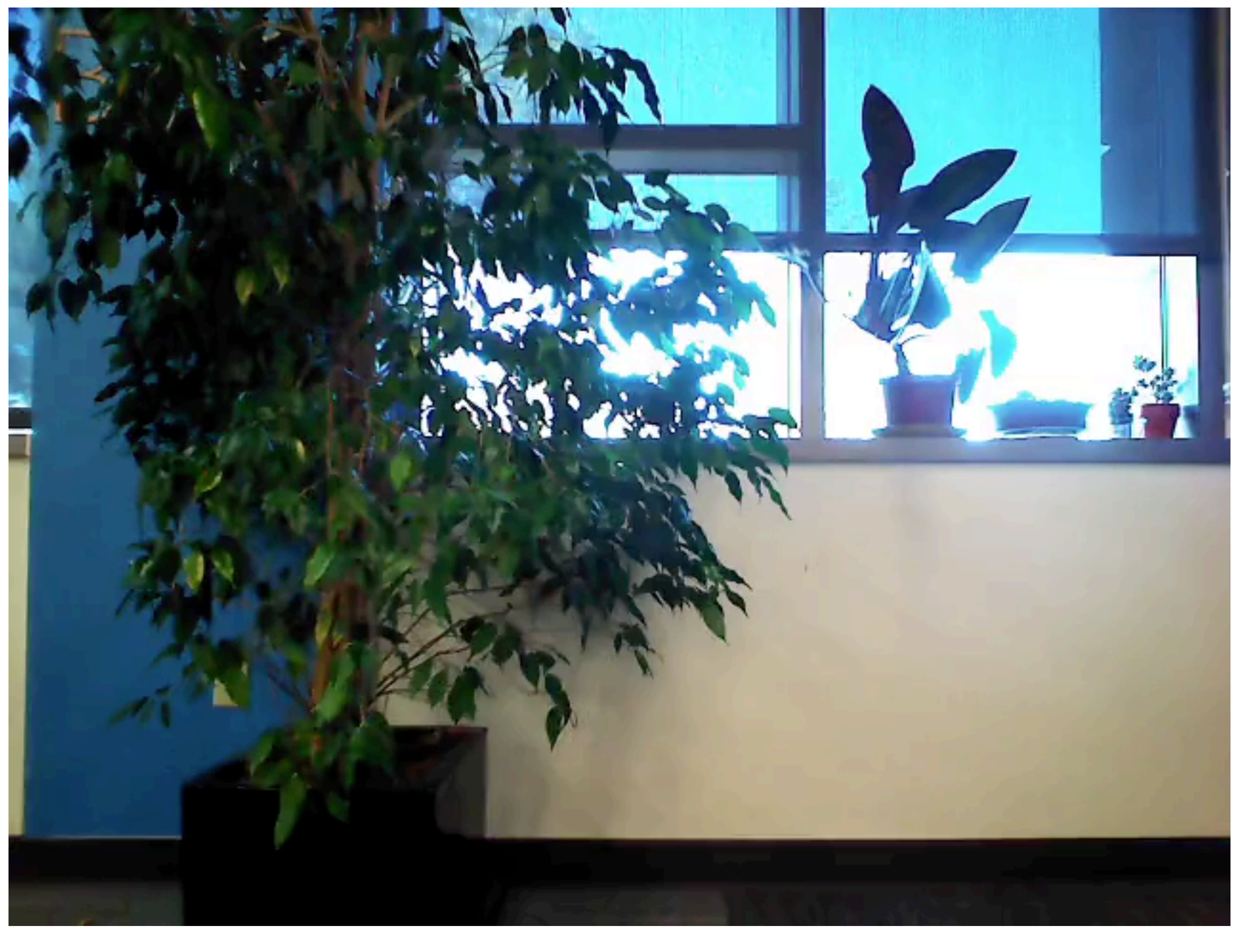


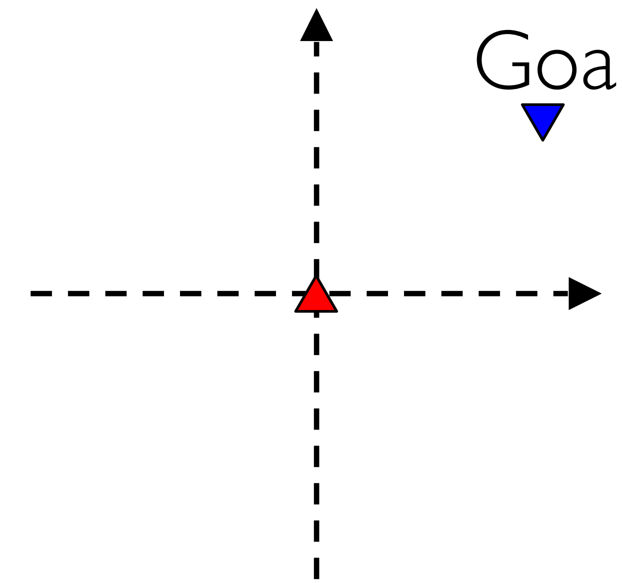
6DOF Pose



Grasp Motion Planning

Robot Navigation





"Go
300 feet North,
400 feet East"

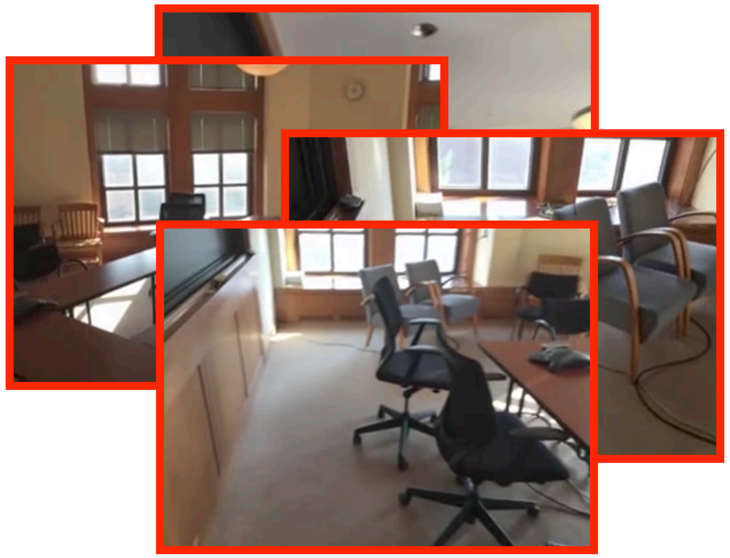
"Go Find a Chair"

Robot with a first person camera

Dropped into a novel environment

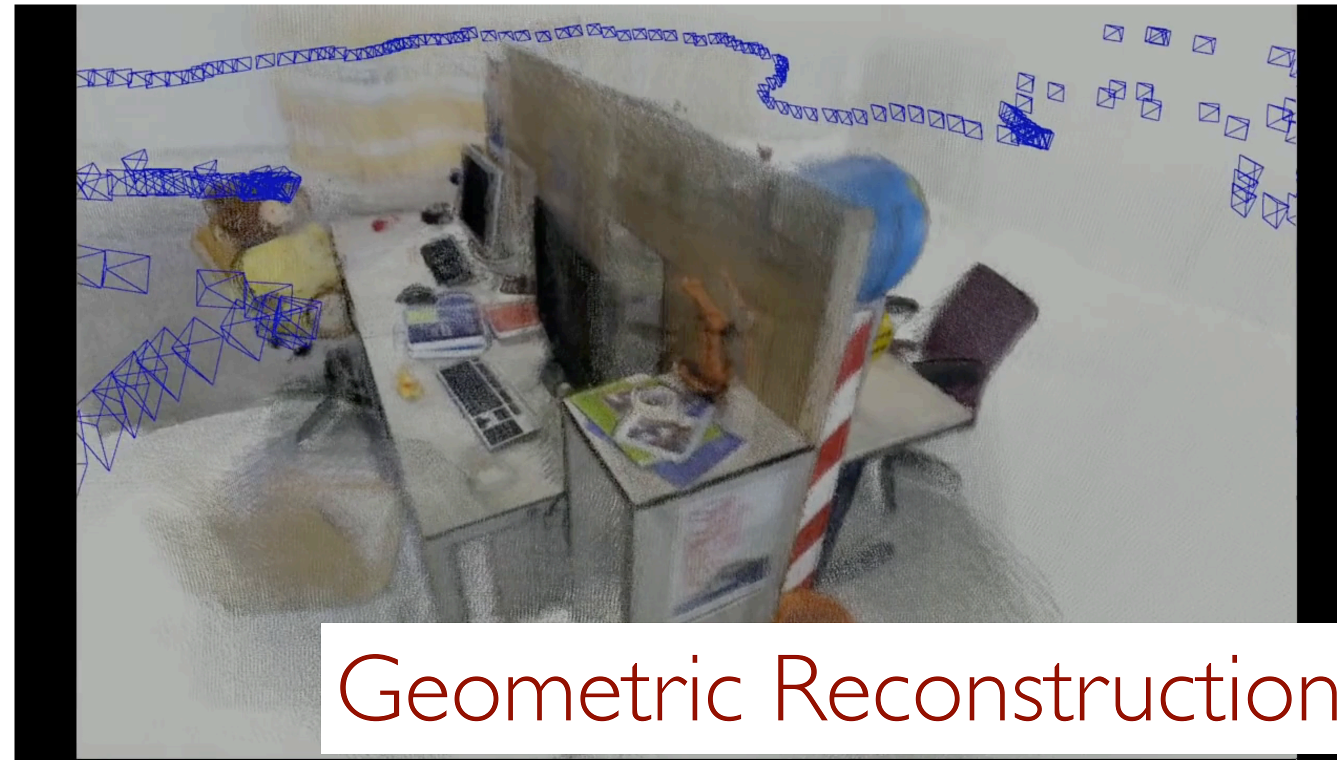
Navigate around

Observations → State Estimation → Planning → Low-level Controller → Control

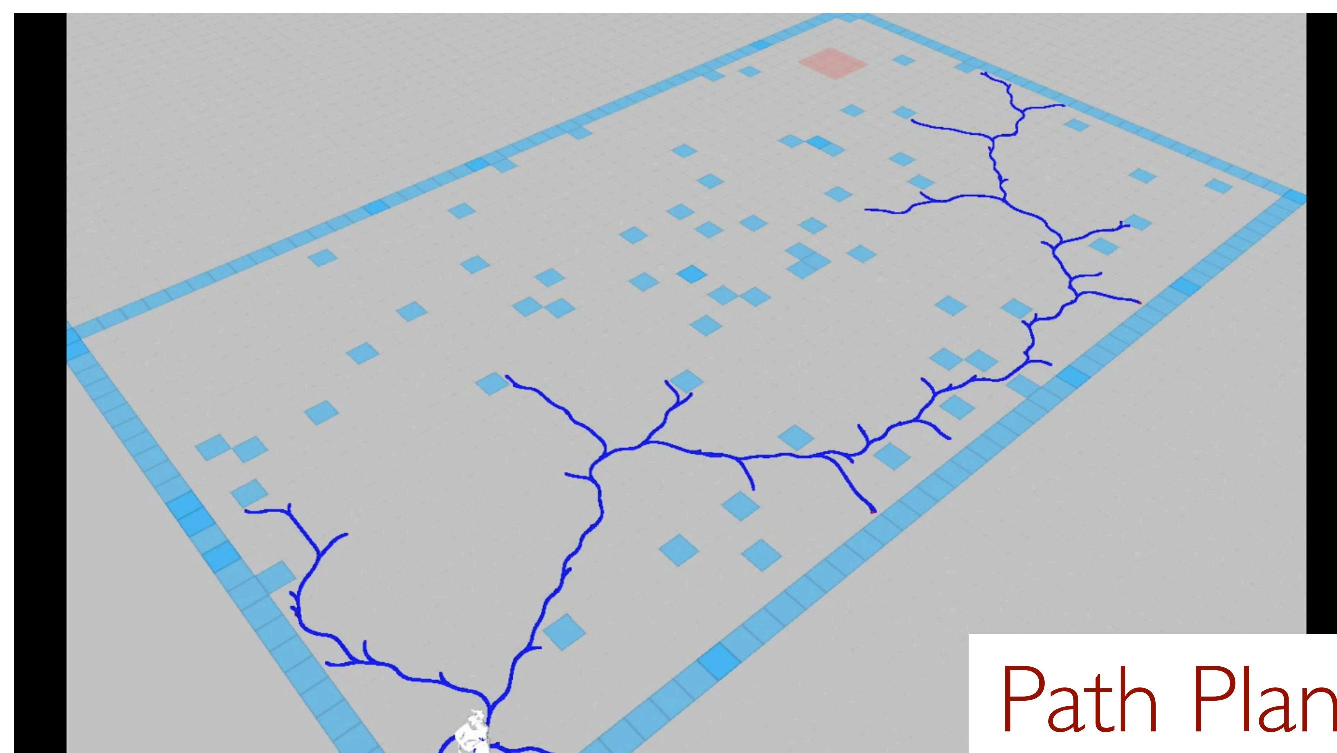


Observed Images

Mapping



Planning



Hartley and Zisserman. 2000. Multiple View Geometry in Computer Vision

Thrun, Burgard, Fox. 2005. Probabilistic Robotics

Canny. 1988. The complexity of robot motion planning.

Kavraki et al. RA1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces.

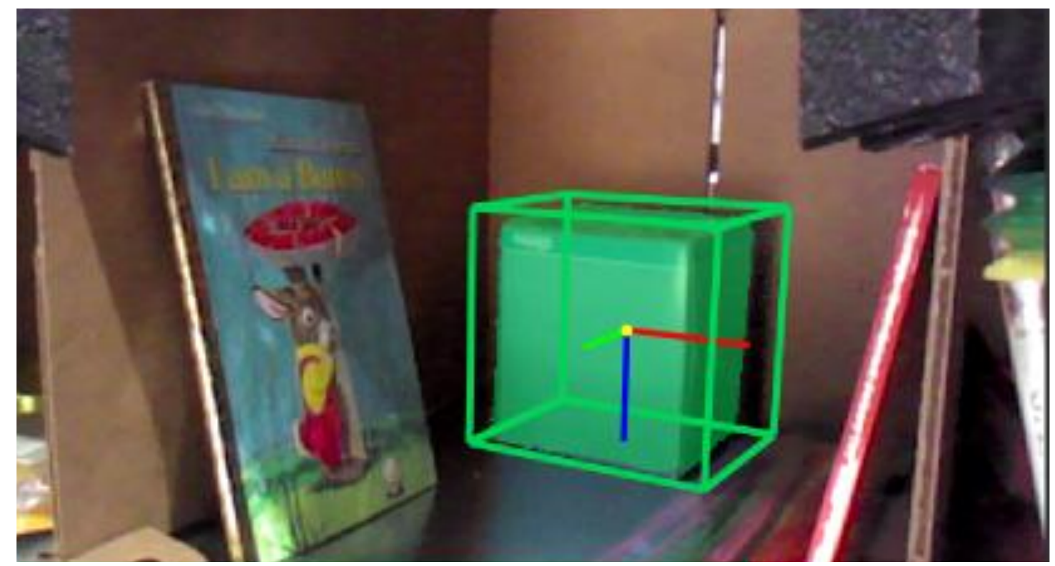
Lavalle and Kuffner. 2000. Rapidly-exploring random trees: Progress and prospects.

Video Credits: Mur-Artal et al., Palmieri et al.

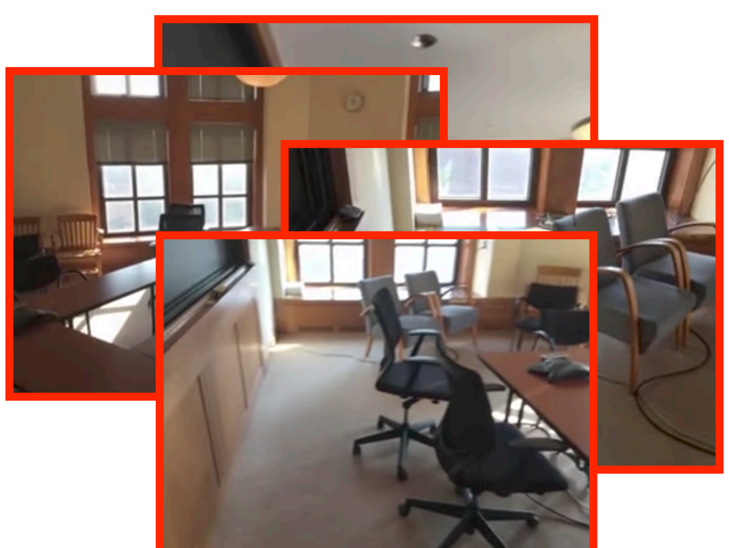
Typical Robotics Pipeline

Observations → State Estimation → Planning → Low-level Controller → Control

Observed Images



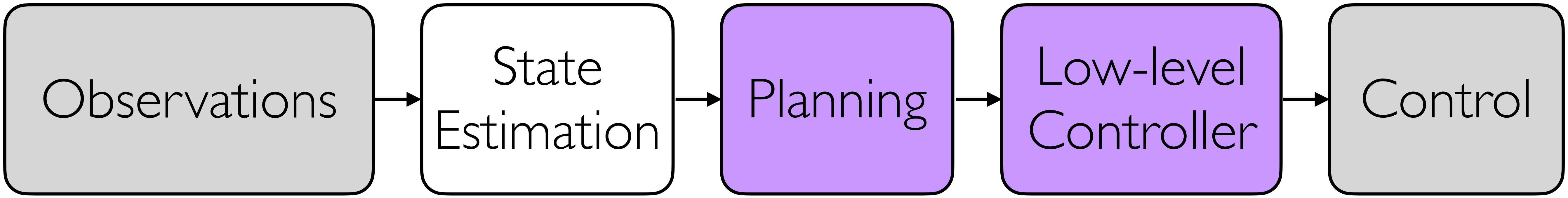
6DOF Pose



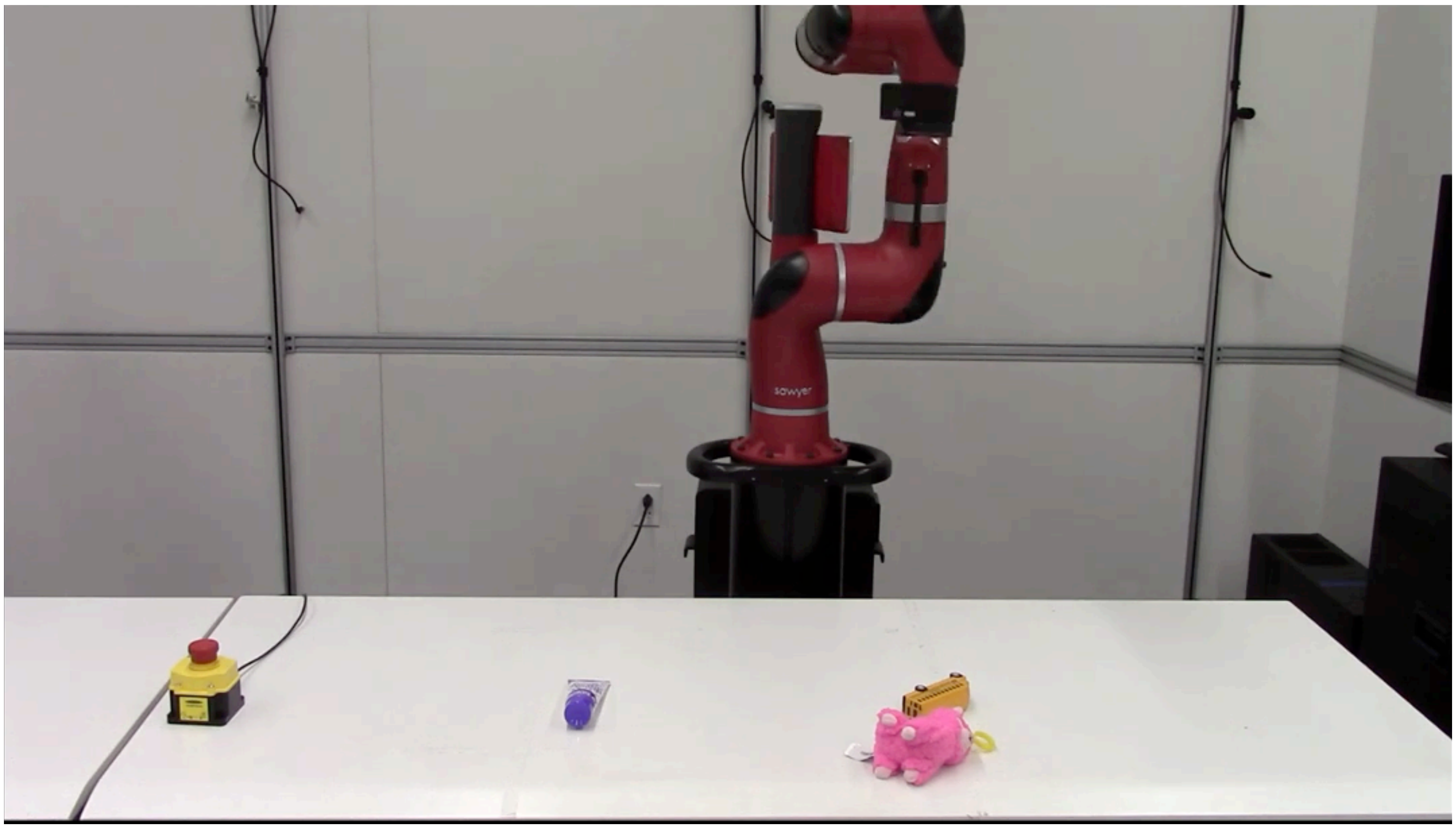
Observed Images

Geometric or Semantic Maps

Typical Robotics Pipeline



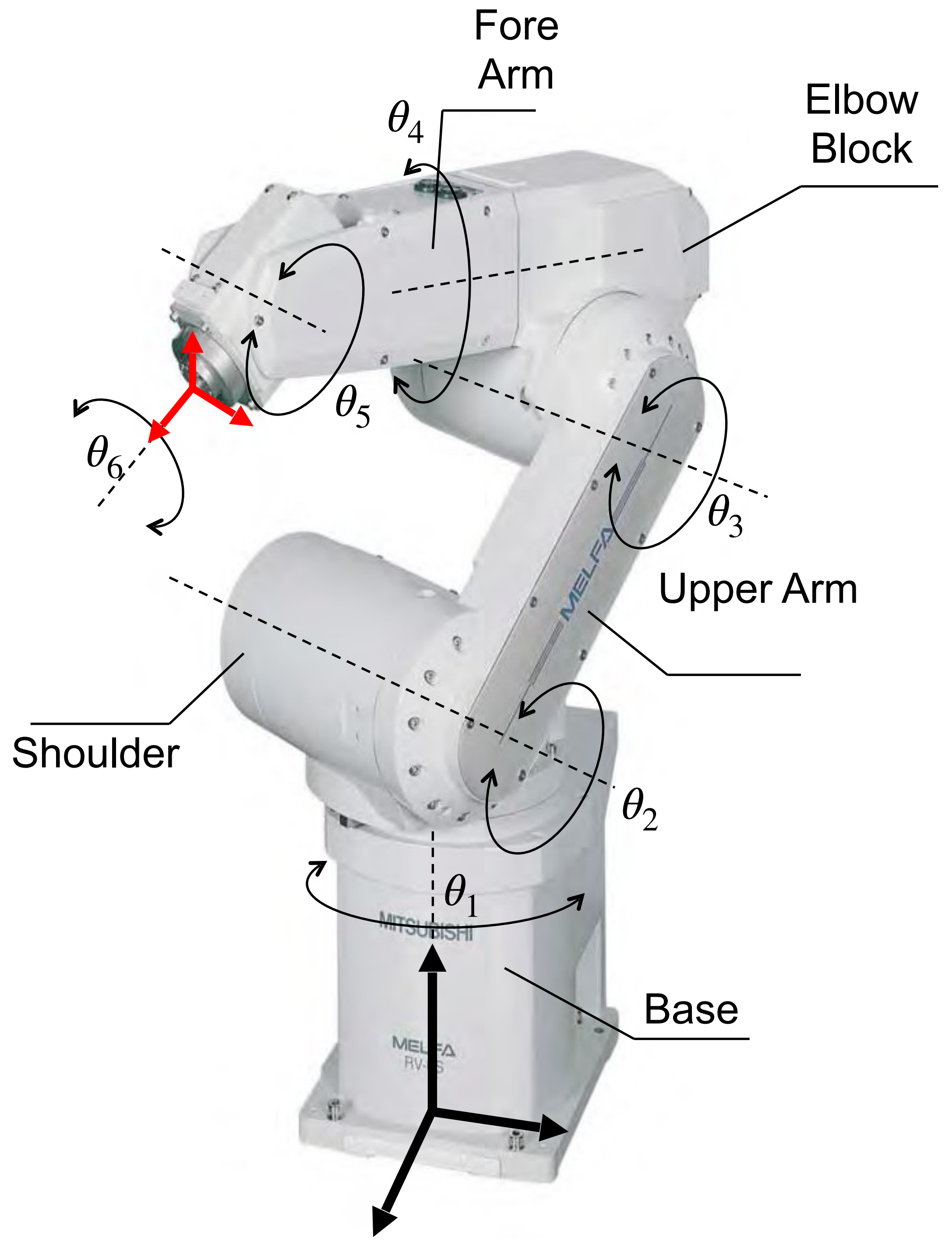
Understand how to move a robot



Video from Deepak Pathak.

Terminology

- Link
- Joint
- End Effector
- Base
- Sensors



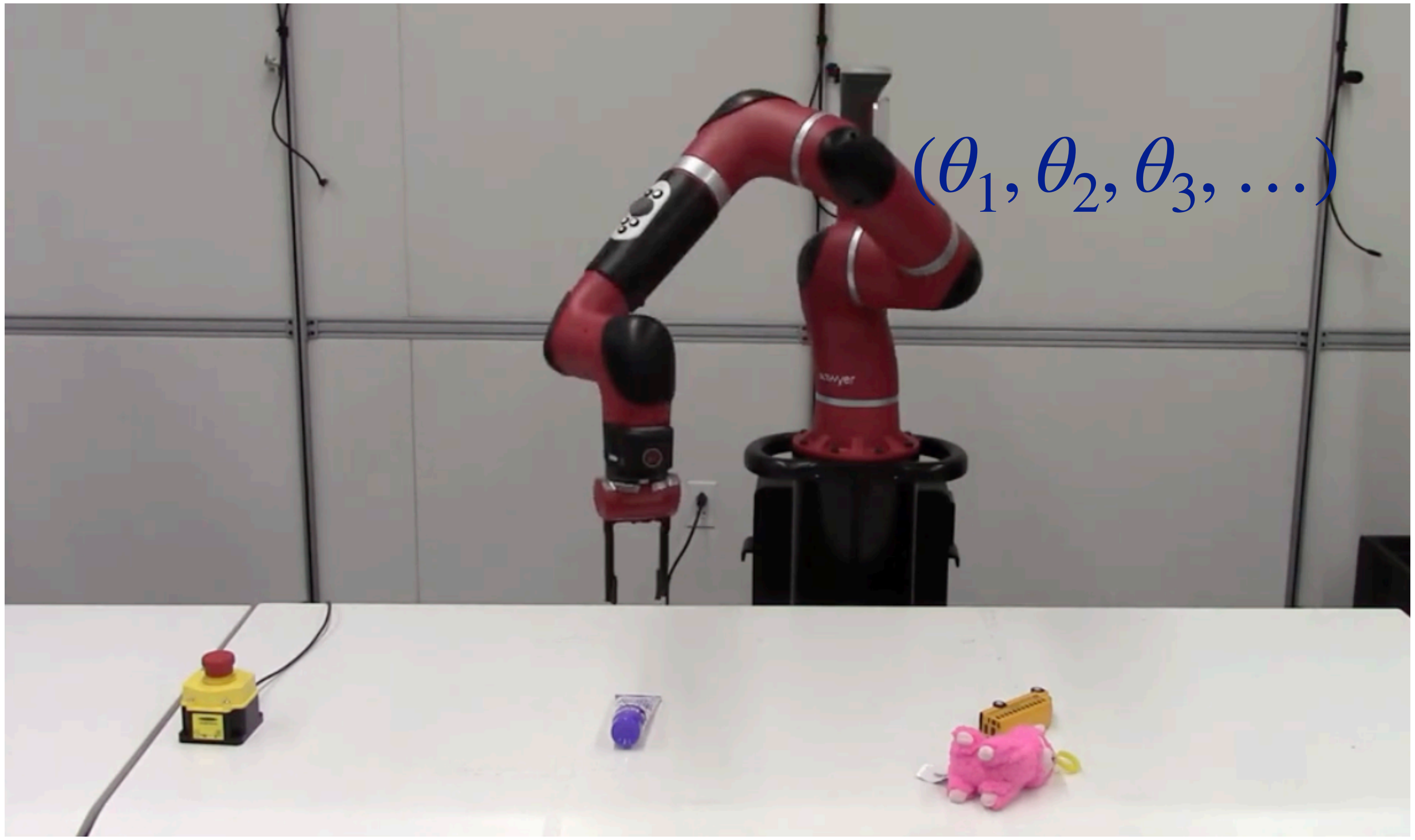
Slide from Dhiraj Gandhi.

Spaces

Work Space

Configuration Space

Task Space

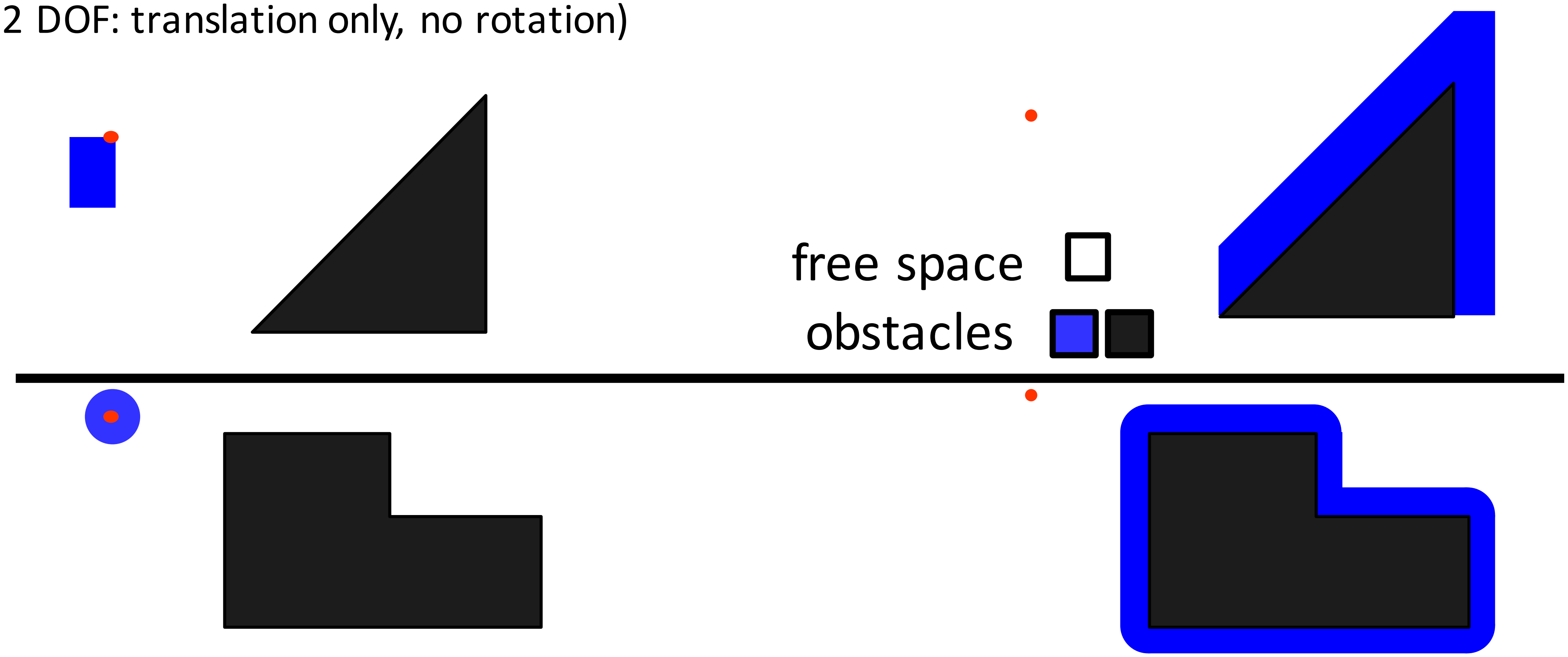


Configuration Space

obstacles → configuration space obstacles

Workspace *Configuration Space*

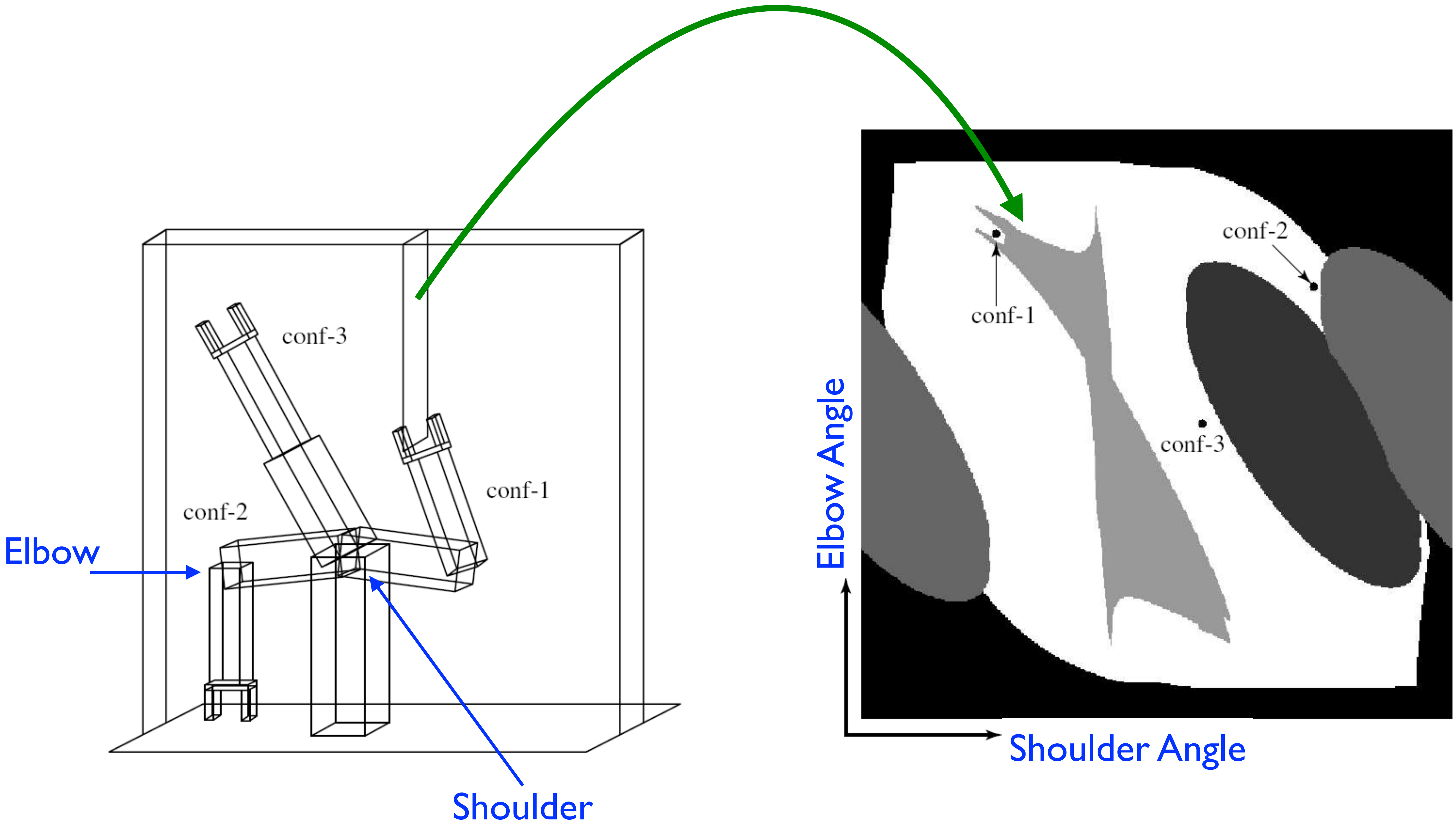
(2 DOF: translation only, no rotation)



Slide from Pieter Abbeel.

Configuration Space

Another Example

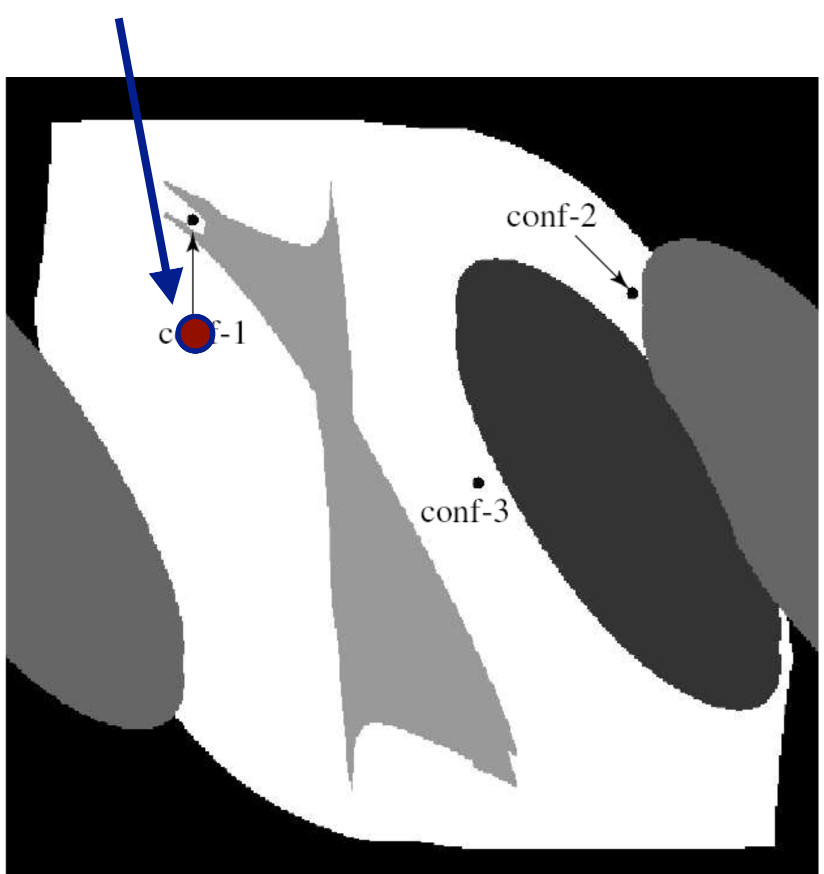


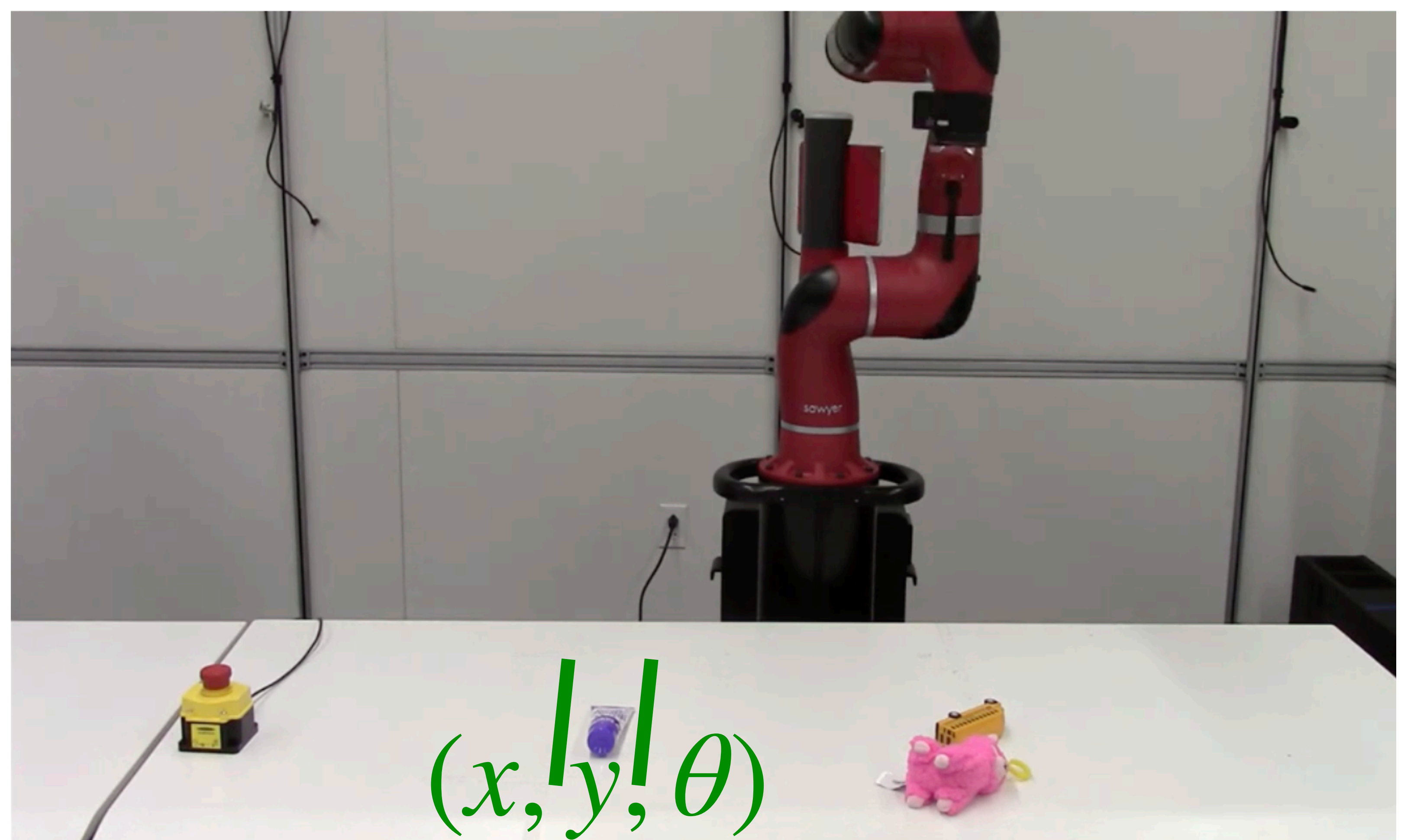
Slide from Pieter Abbeel.

How to move your robot?

1. Task space to Configuration space

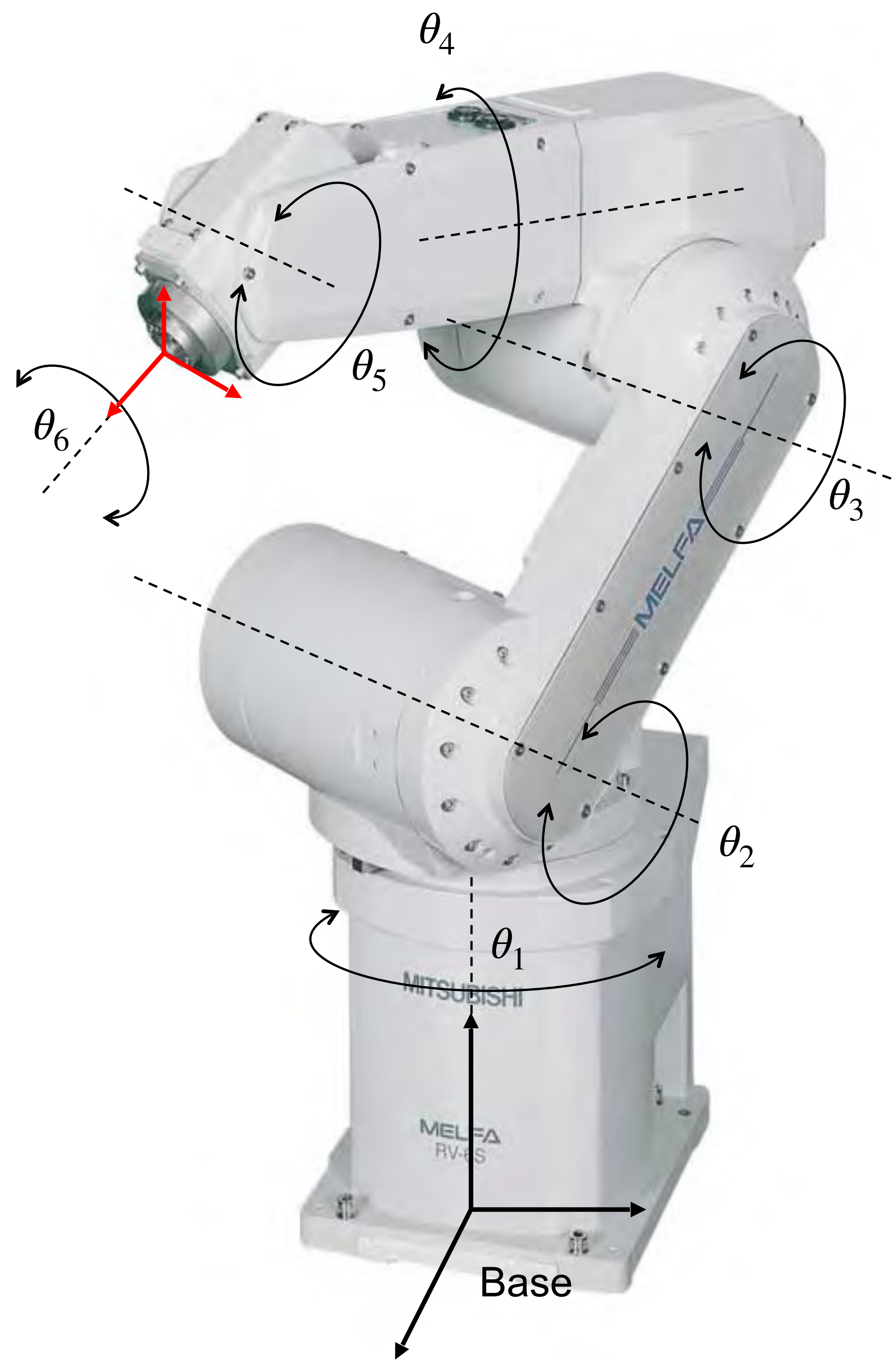
Initial configuration





Configuration Space to Task Space

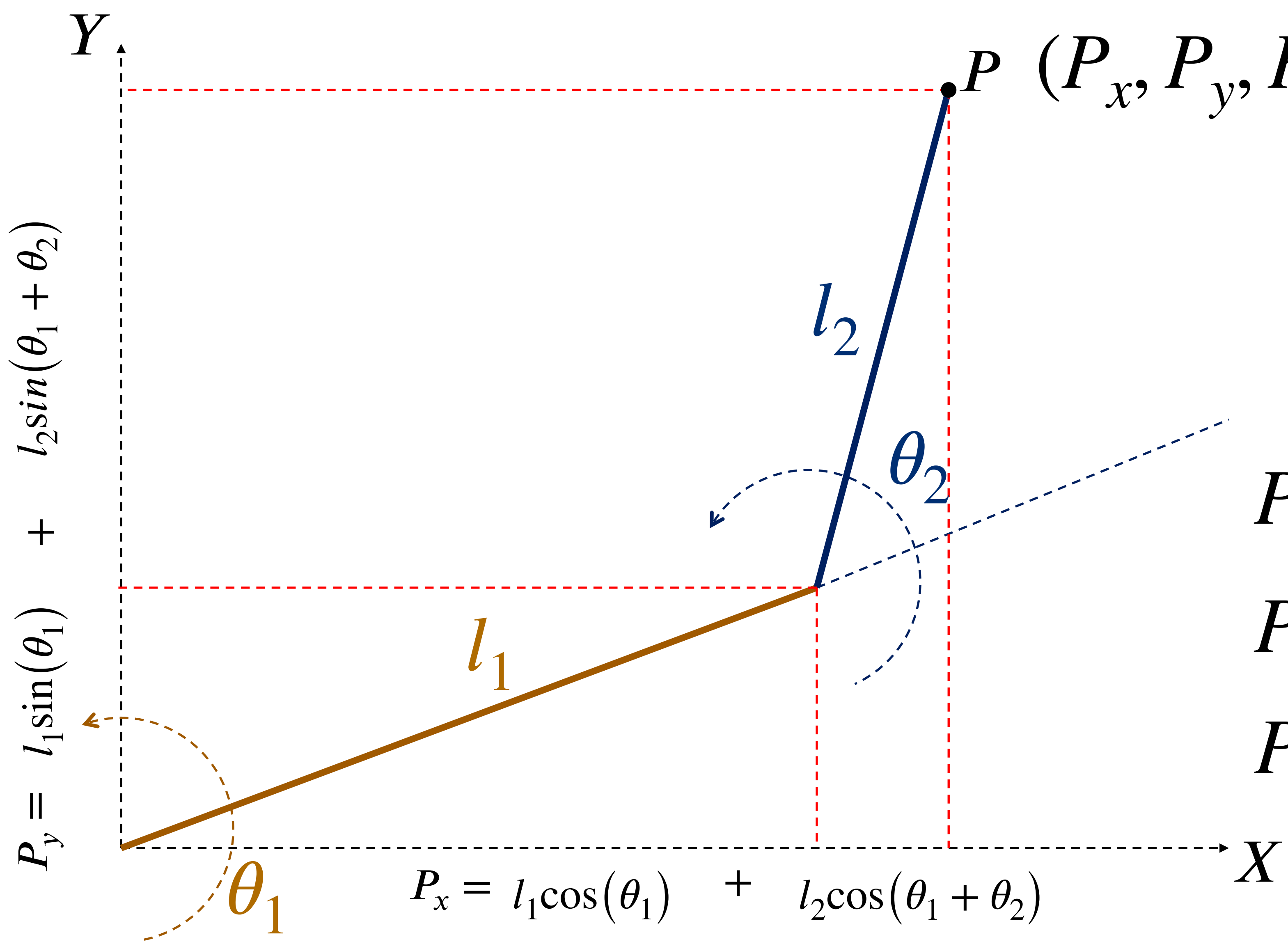
Forward Kinematics



Slide from Dhiraj Gandhi.

Configuration Space to Task Space

Forward Kinematics



$$P_x = l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2)$$

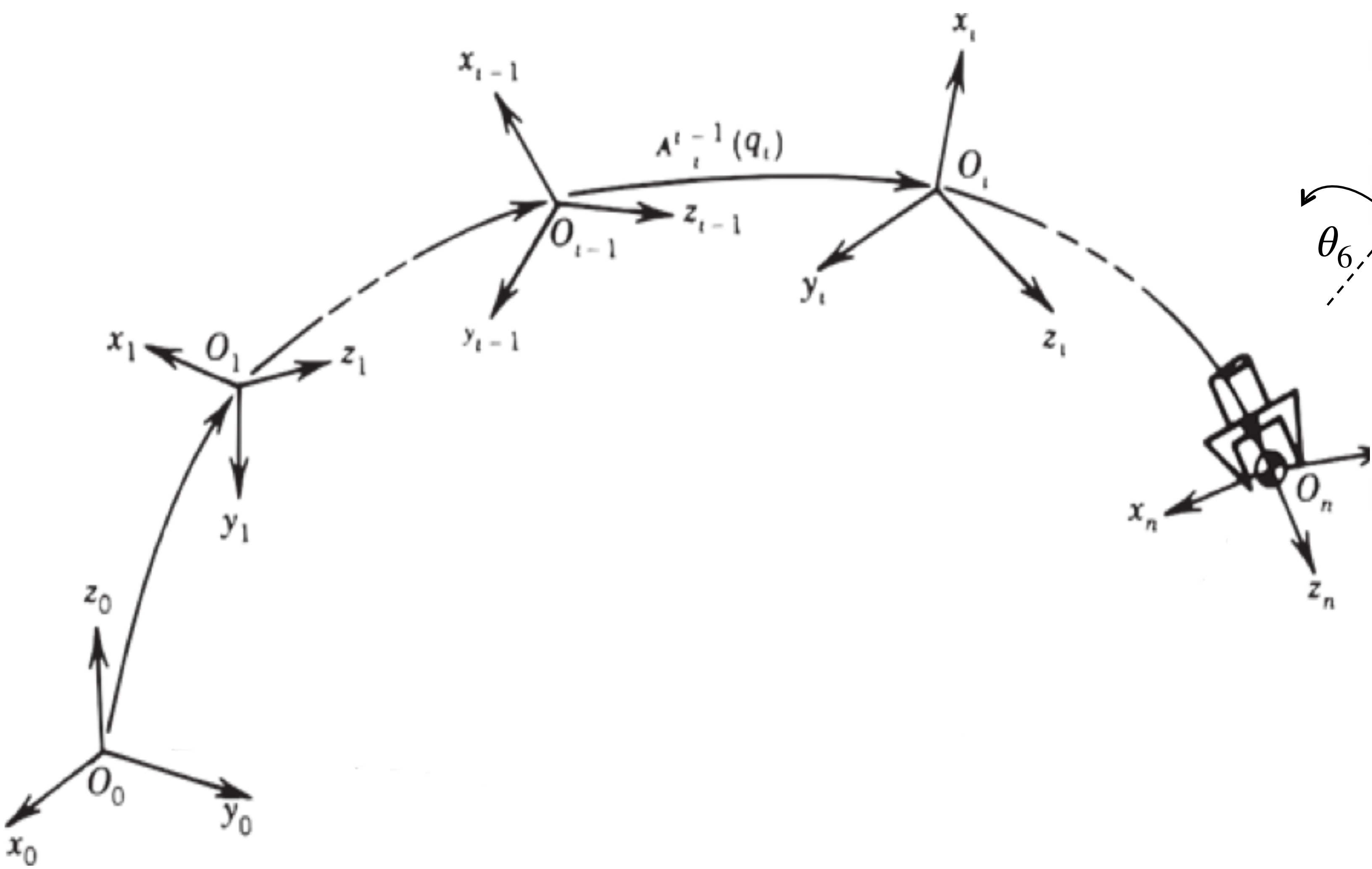
$$P_y = l_2 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2)$$

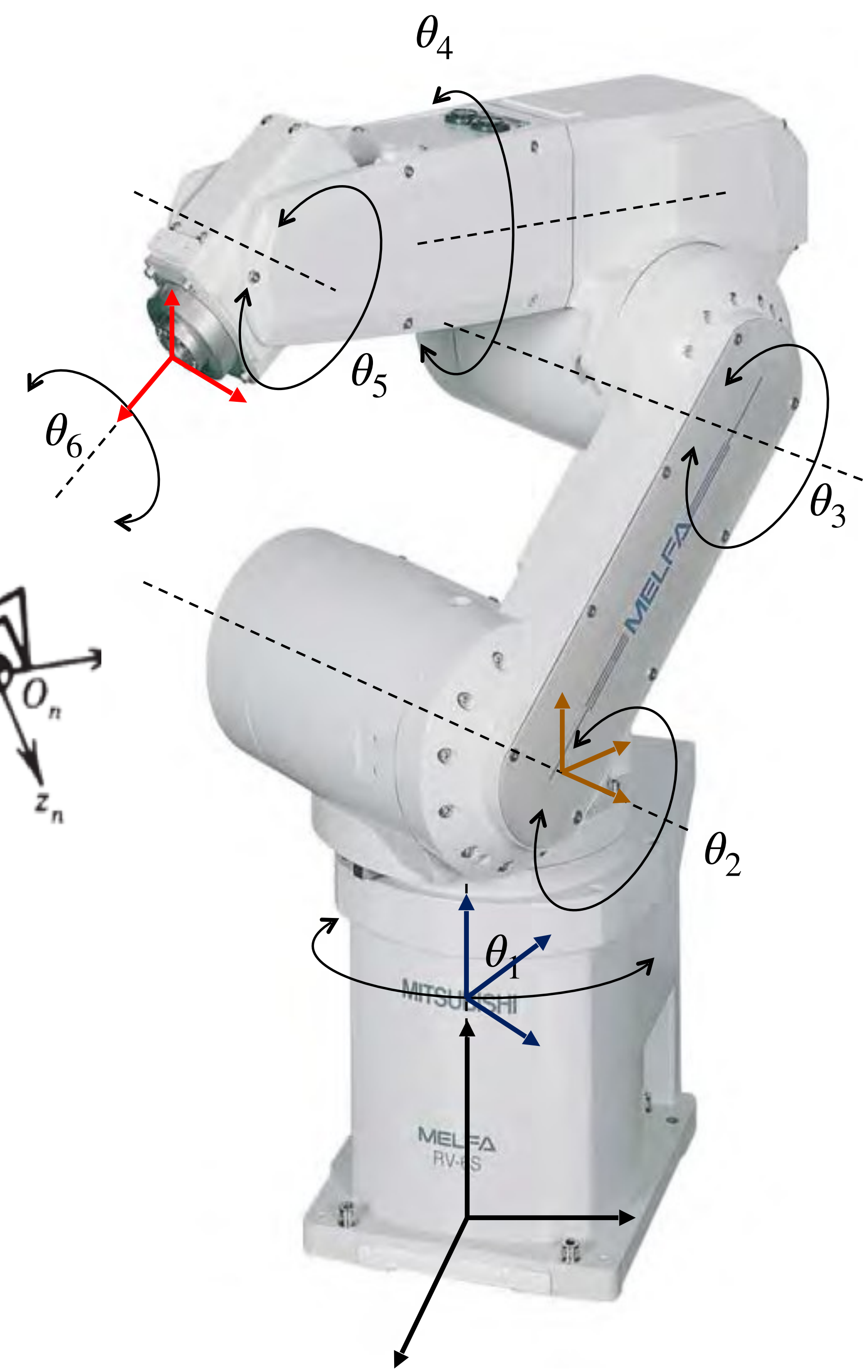
$$P_\theta = \theta_1 + \theta_2$$

Slide from Dhiraj Gandhi.

Configuration Space to Task Space

Forward Kinematics

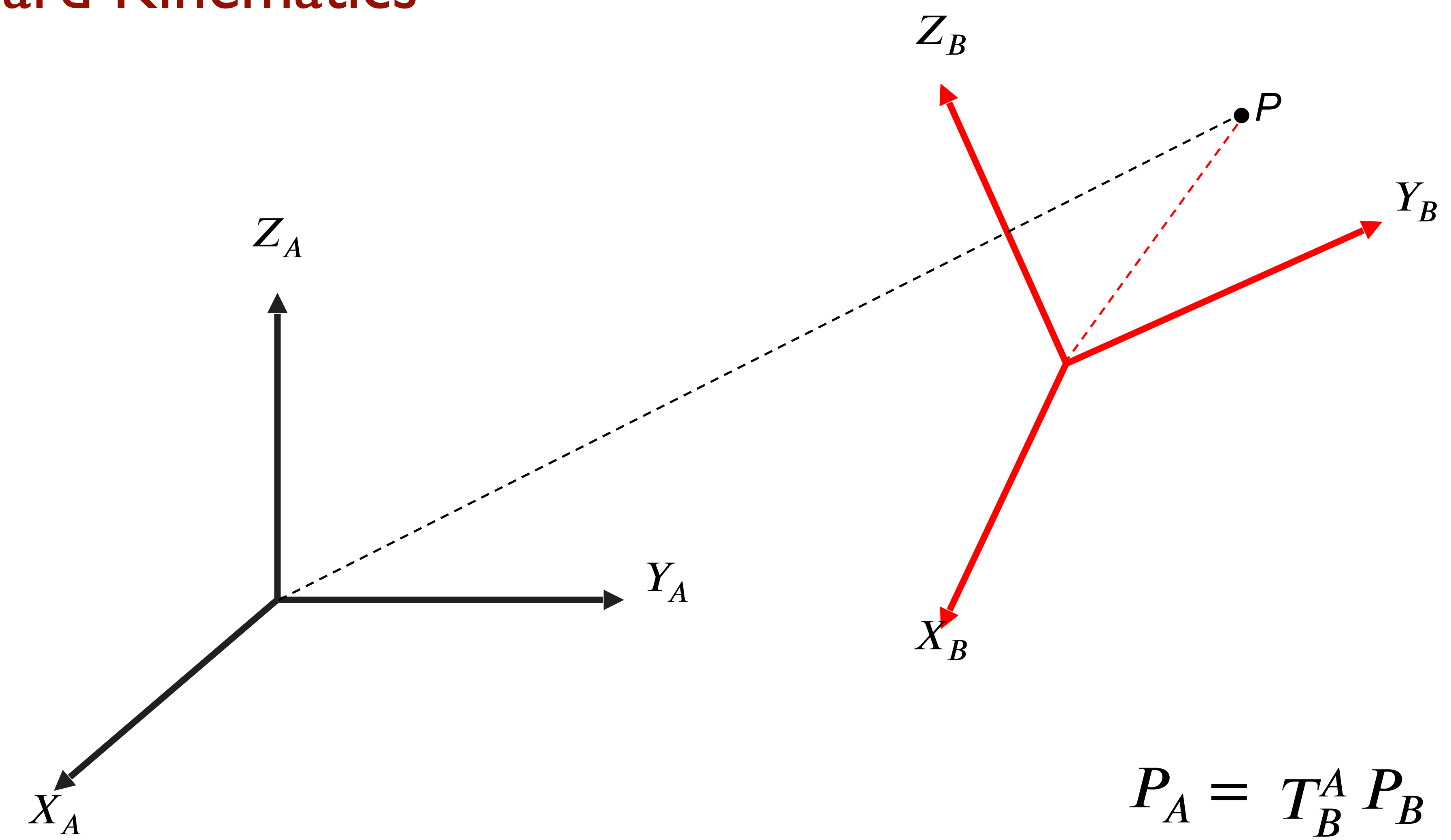




Slide from Dhiraj Gandhi.

Configuration Space to Task Space

Forward Kinematics



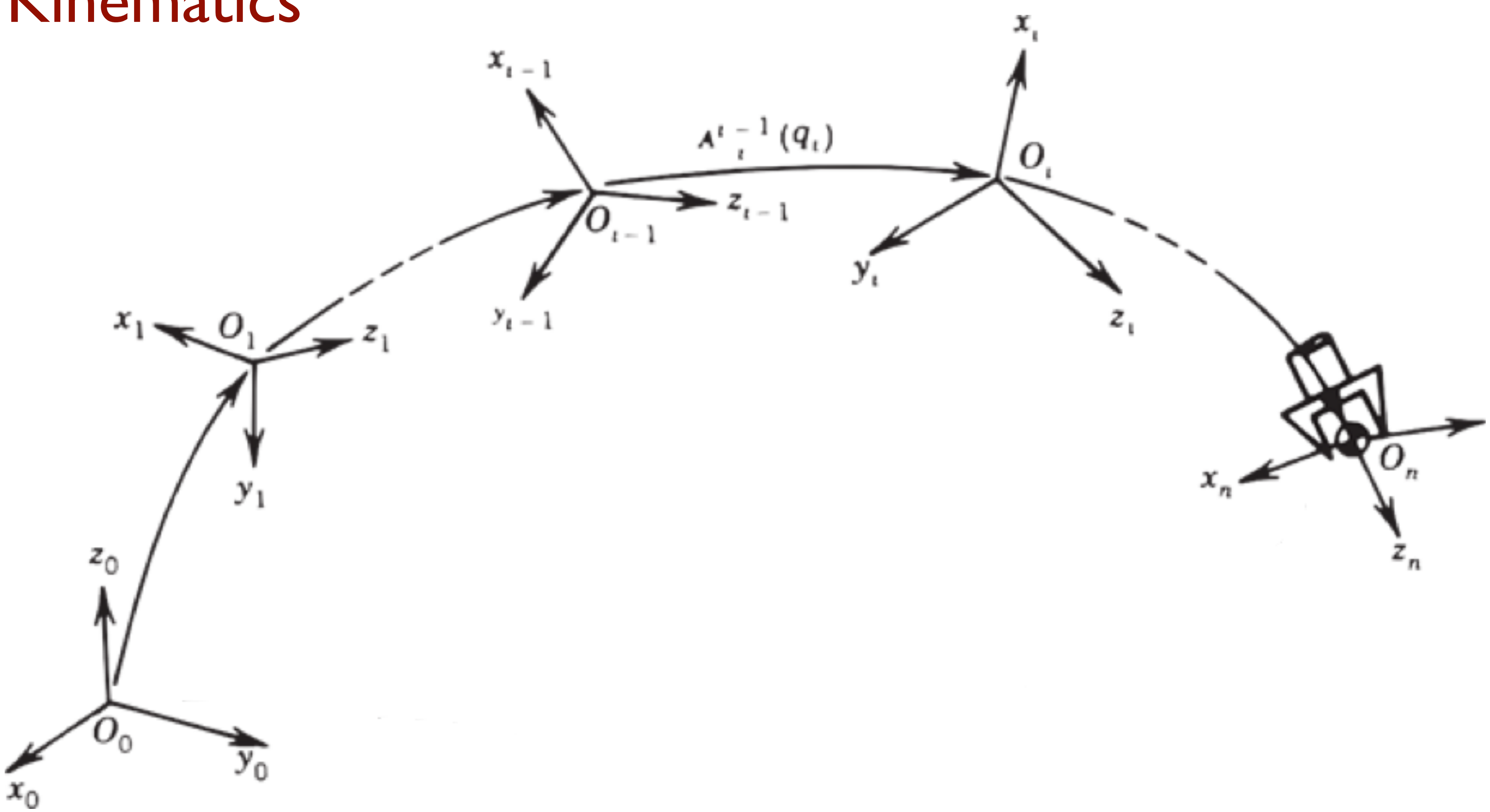
$$P_A = T_B^A P_B$$

$$T_B^A = \begin{bmatrix} r_{11} & r_{21} & r_{31} & \Delta x \\ r_{12} & r_{22} & r_{32} & \Delta y \\ r_{13} & r_{23} & r_{33} & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Slide from Dhiraj Gandhi.

Configuration Space to Task Space

Forward Kinematics

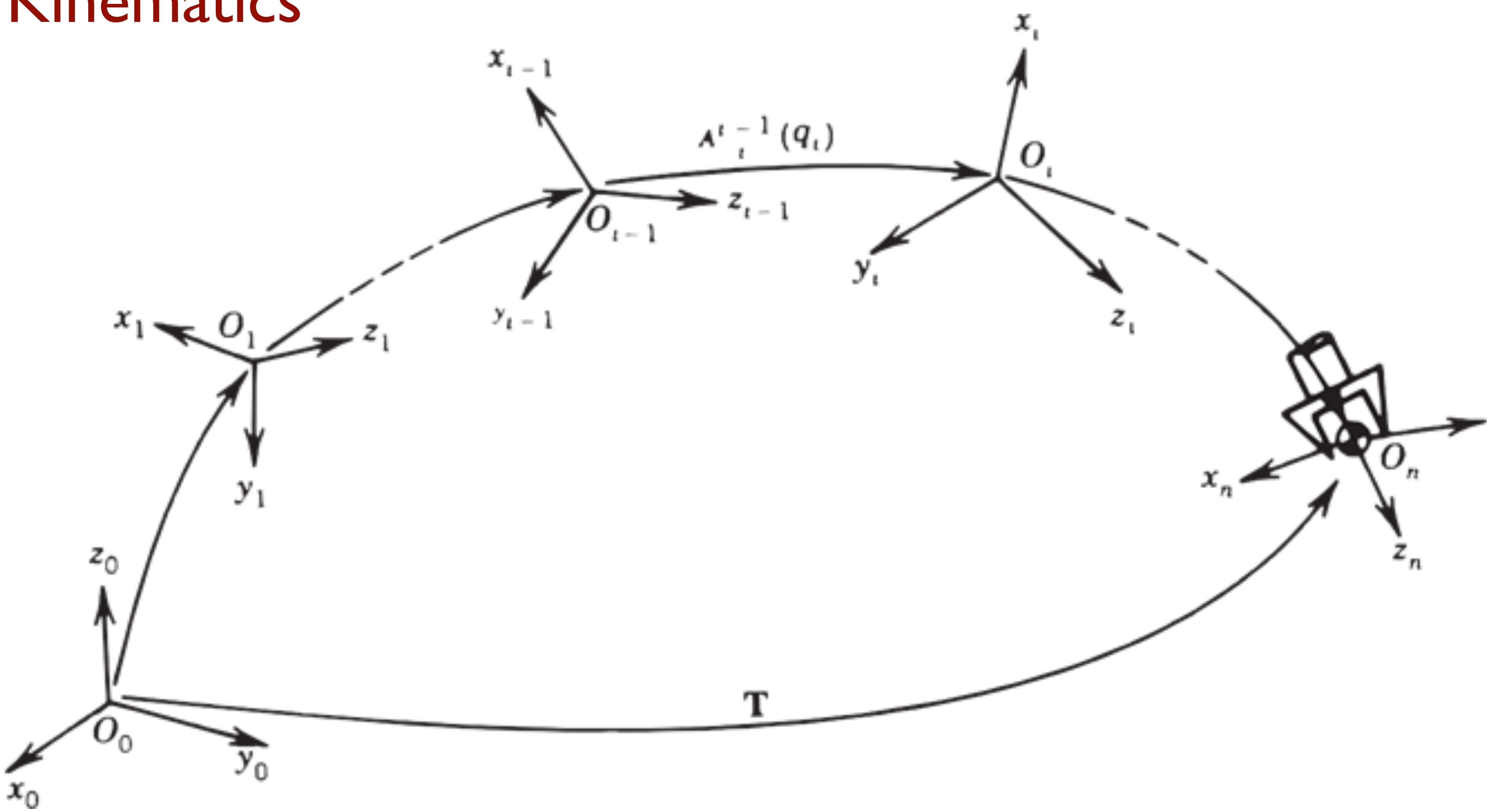


$$T_1^0(\theta_1)\ T_2^1(\theta_2)\ldots \quad T_{n-1}^{n-2}(\theta_n) \quad T_n^{n-1}(\theta_n)$$

Slide from Dhiraj Gandhi.

Configuration Space to Task Space

Forward Kinematics



$$T = T_1^0(\theta_1)\, T_2^1(\theta_2)\ldots \quad T_{n-1}^{n-2}(\theta_n) \quad T_n^{n-1}(\theta_n)$$

Maps configuration space to work space

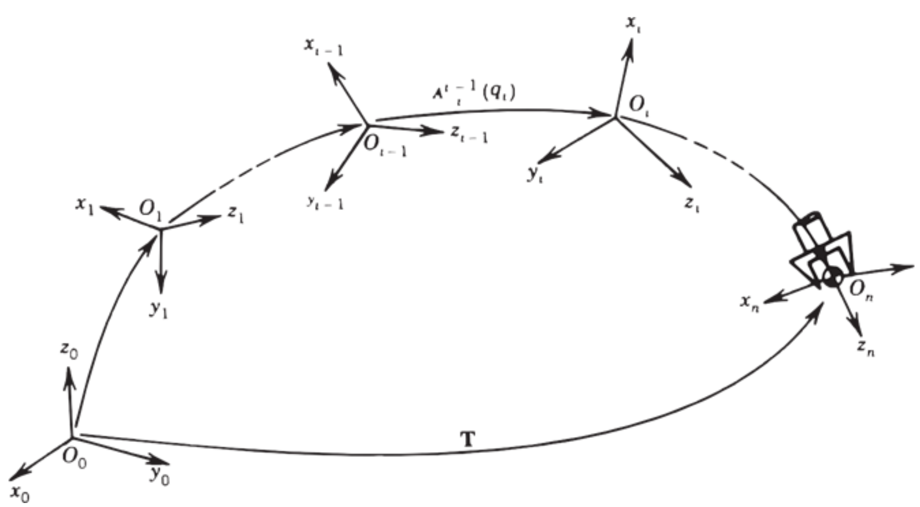
$$= \begin{bmatrix} r_{11} & r_{21} & r_{31} & \Delta x \\ r_{12} & r_{22} & r_{32} & \Delta y \\ r_{13} & r_{23} & r_{33} & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x = f(\theta) = f(\theta_1, \theta_2, \ldots, \theta_{n-1}, \theta_n)$$

Slide adapted from Dhiraj Gandhi.

Task Space to Configuration Space

Forward Kinematics



$$x = f(\theta)$$

Maps configuration space to work space

Inverse Kinematics

Numerical IK

Solve for θ_d in:

$$x_d - f(\theta_d) = 0$$

Analytical IK

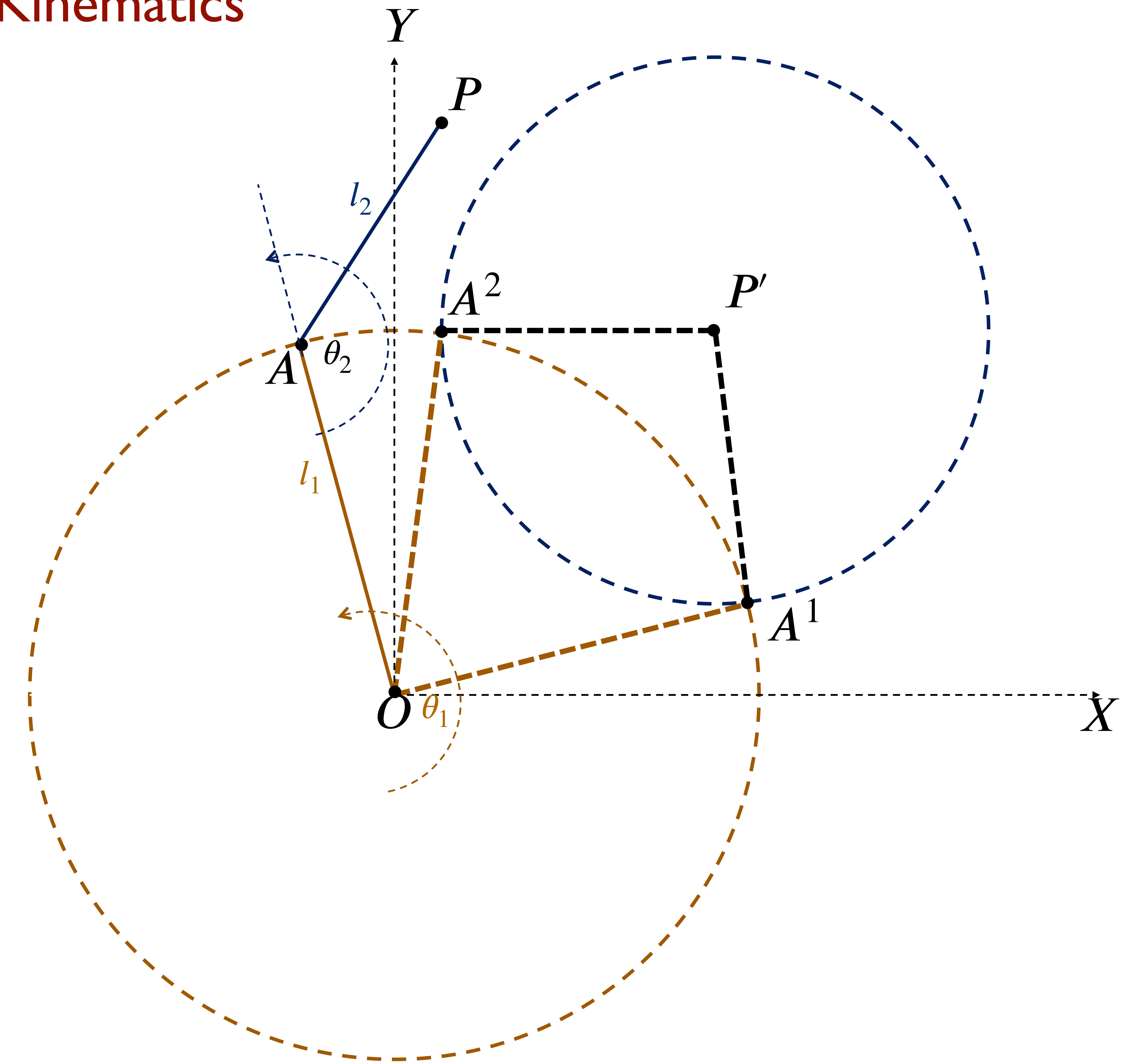
- Robot Specific
- Fast
- Characterize the solution space

Find configuration(s) that map to a given work space point

Slide adapted from Dhiraj Gandhi, Modern Robotics

Task Space to Configuration Space

Analytical Inverse Kinematics



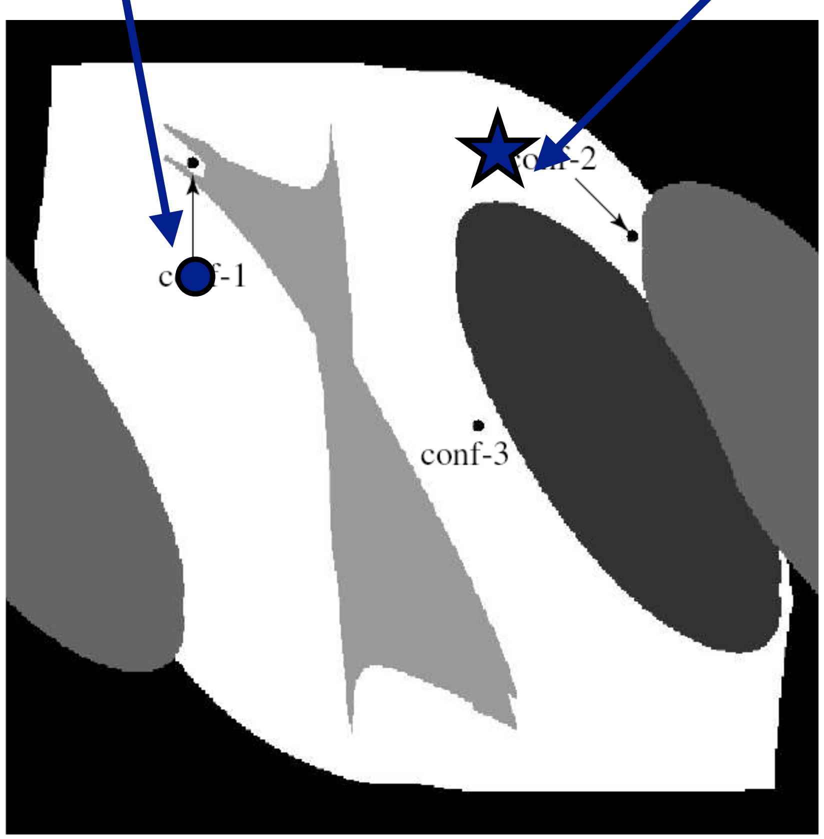
Slide from Dhiraj Gandhi.

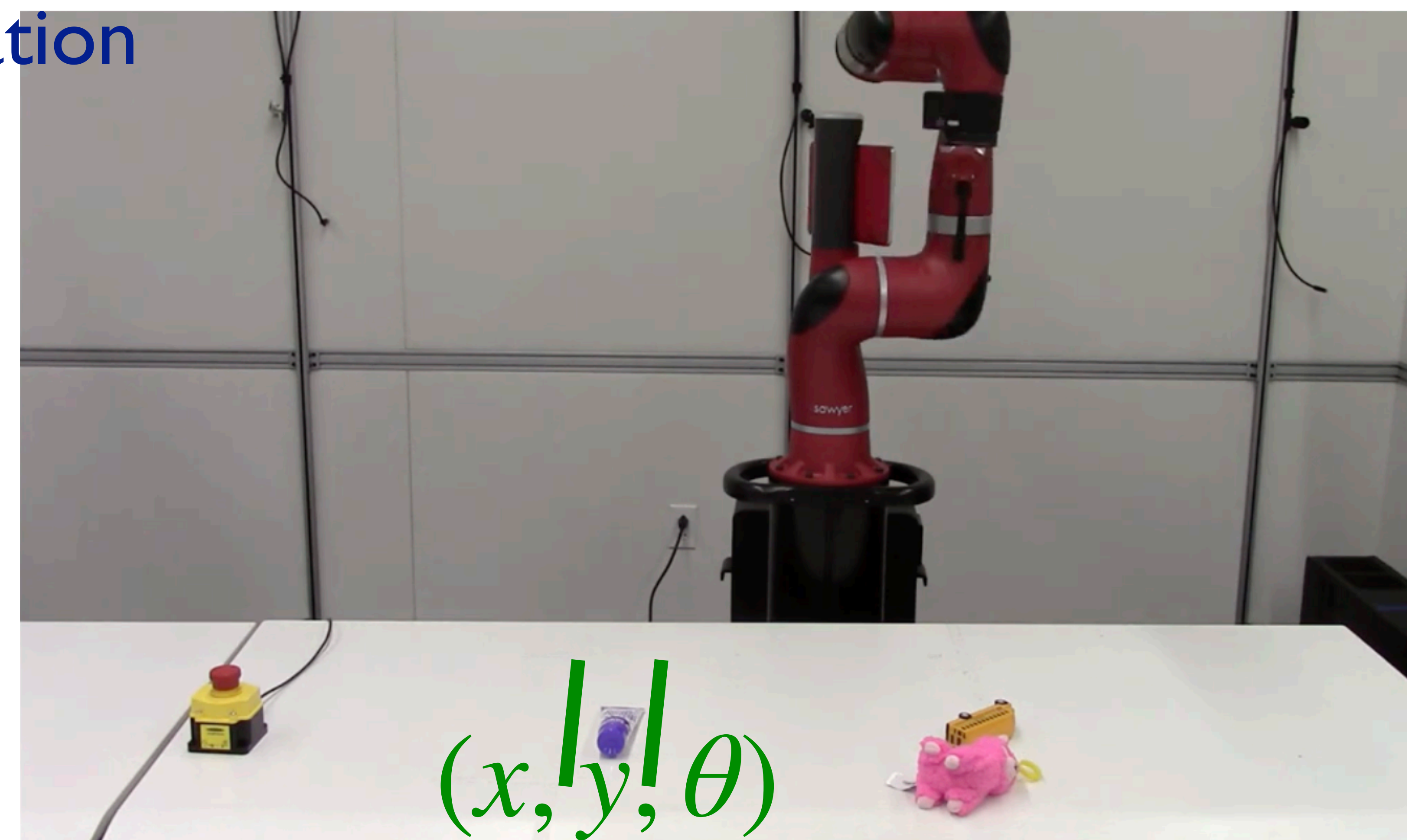
How to move your robot?

1. Task space to Configuration space

Initial configuration

Desired configuration





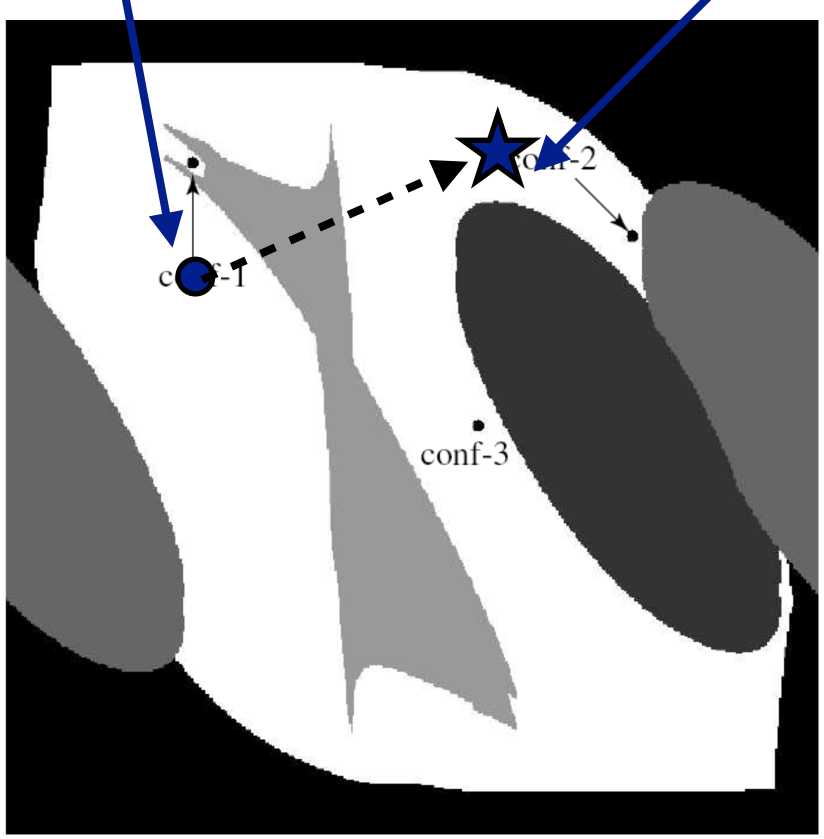
How to move your robot?

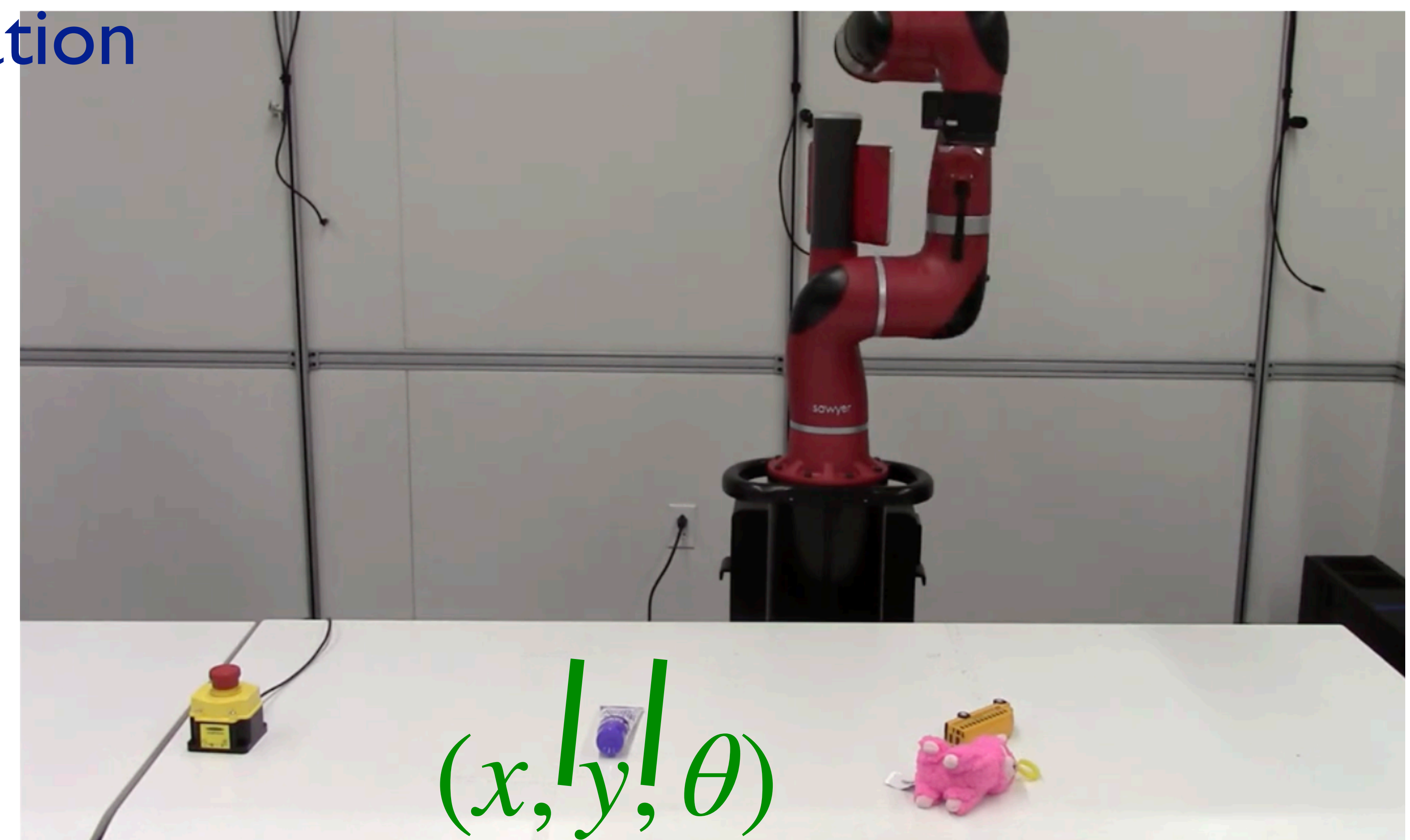
1. Task space to Configuration space

2. Configuration space trajectory

Initial configuration

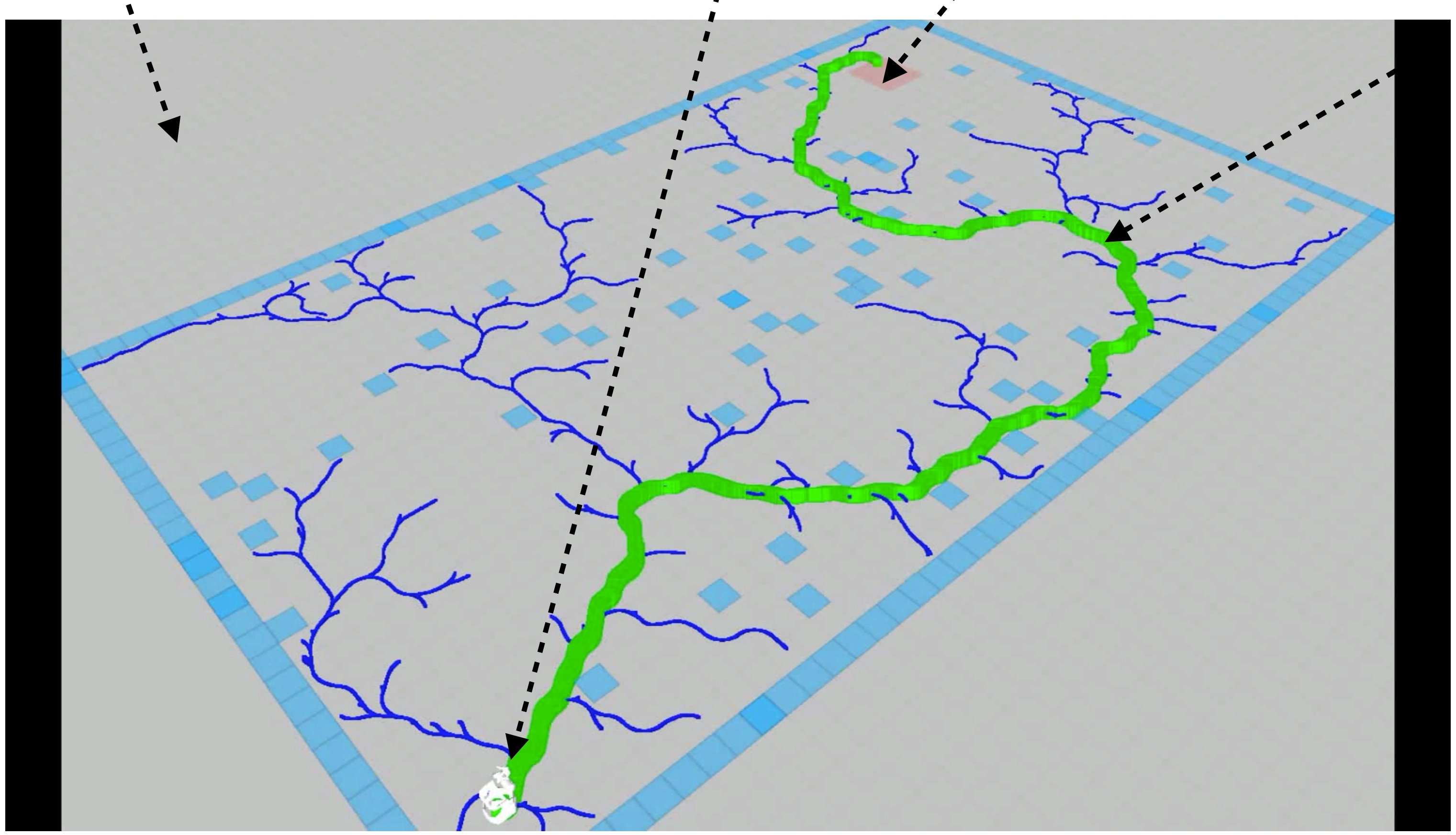
Desired configuration





Path Planning

Configuration Space With Obstacles + Initial Config. + Goal Config. ⟶ Feasible State Trajectory



Picture Credits: Palmieri et al.

Path Planning

1. Complete Methods

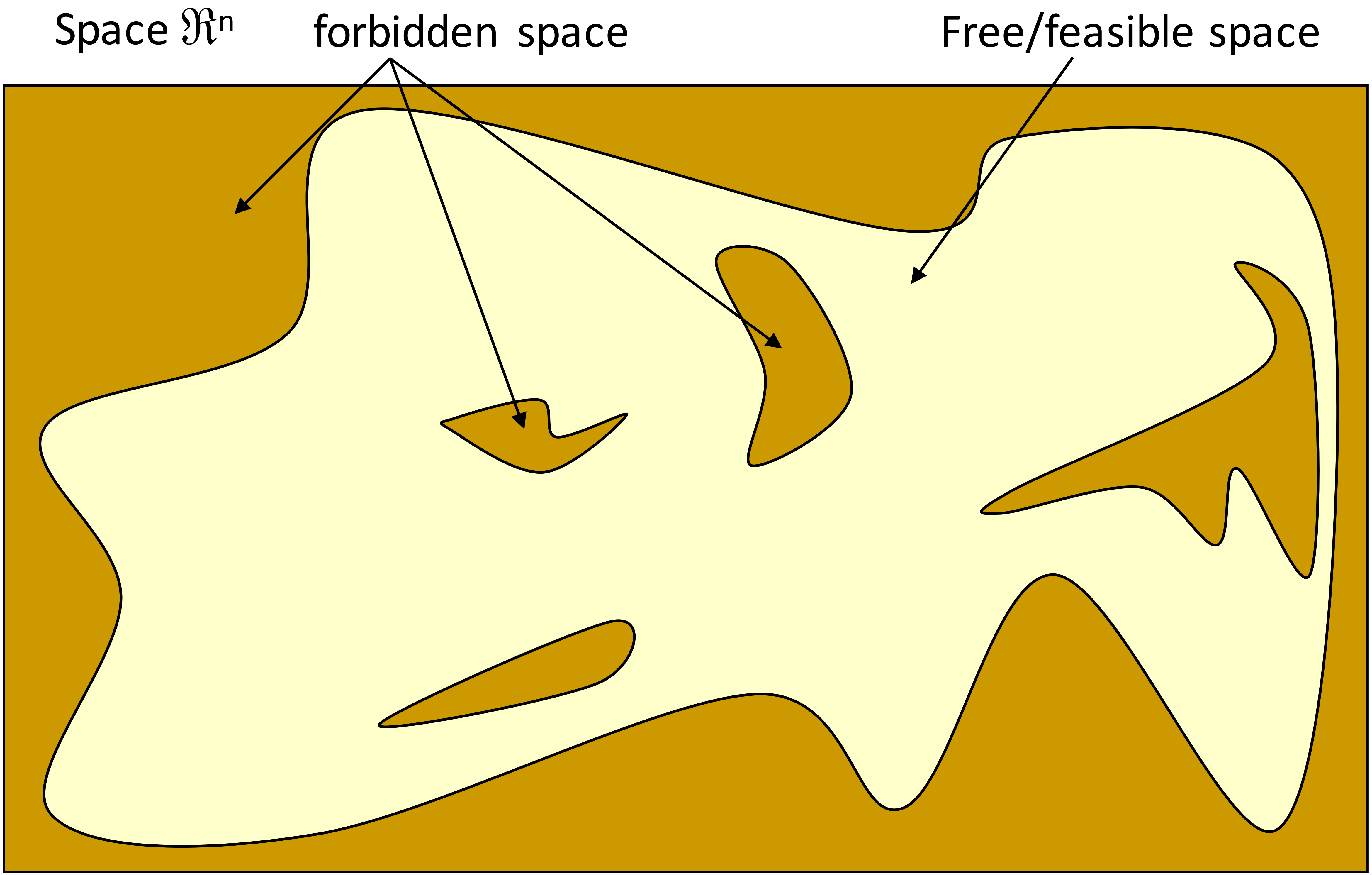
2. Grid Methods

3. Sampling Methods

4. Potential Fields

5. Trajectory Optimization

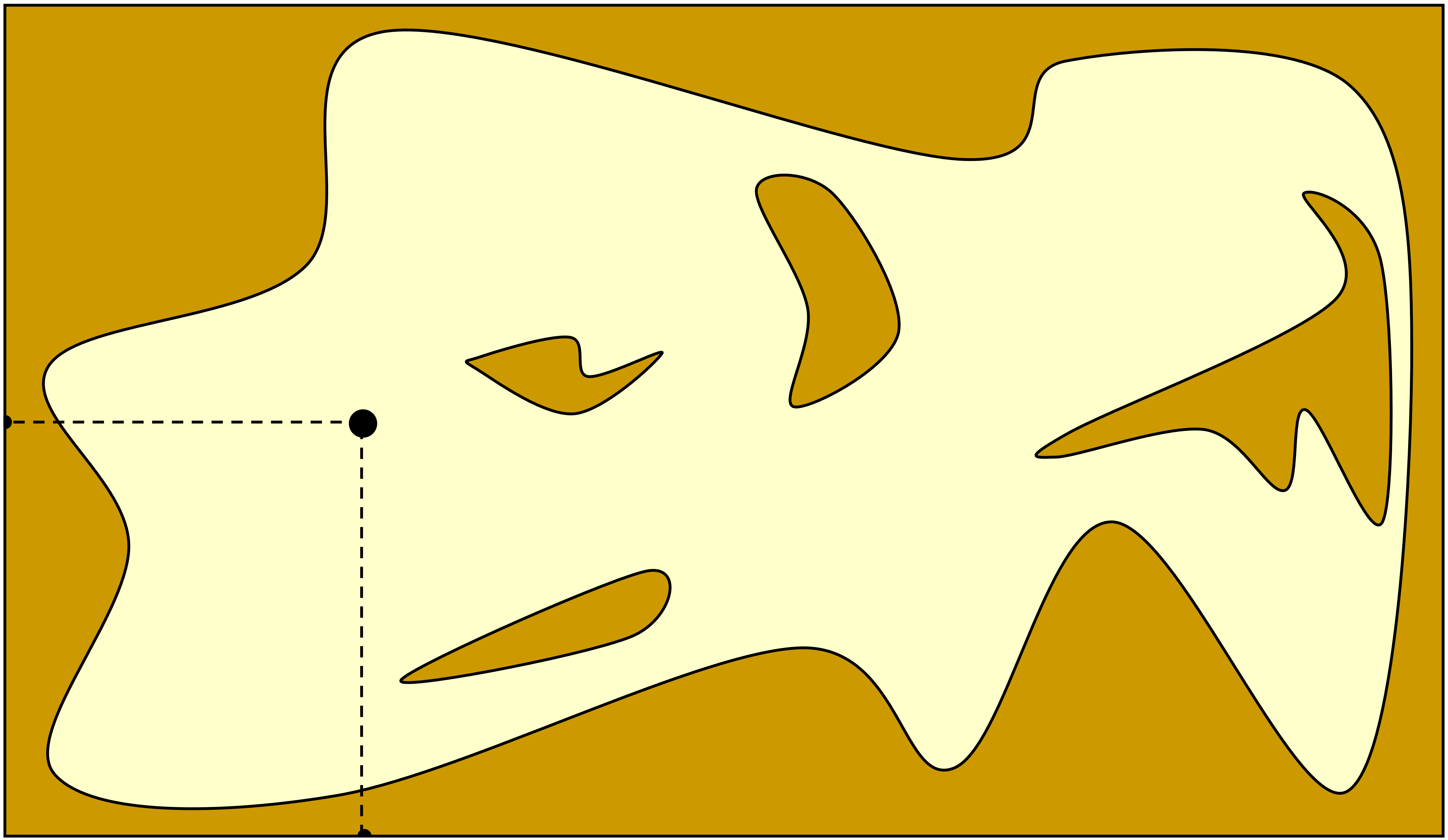
Probabilistic Roadmaps



Slide from Pieter Abbeel.

Probabilistic Roadmaps

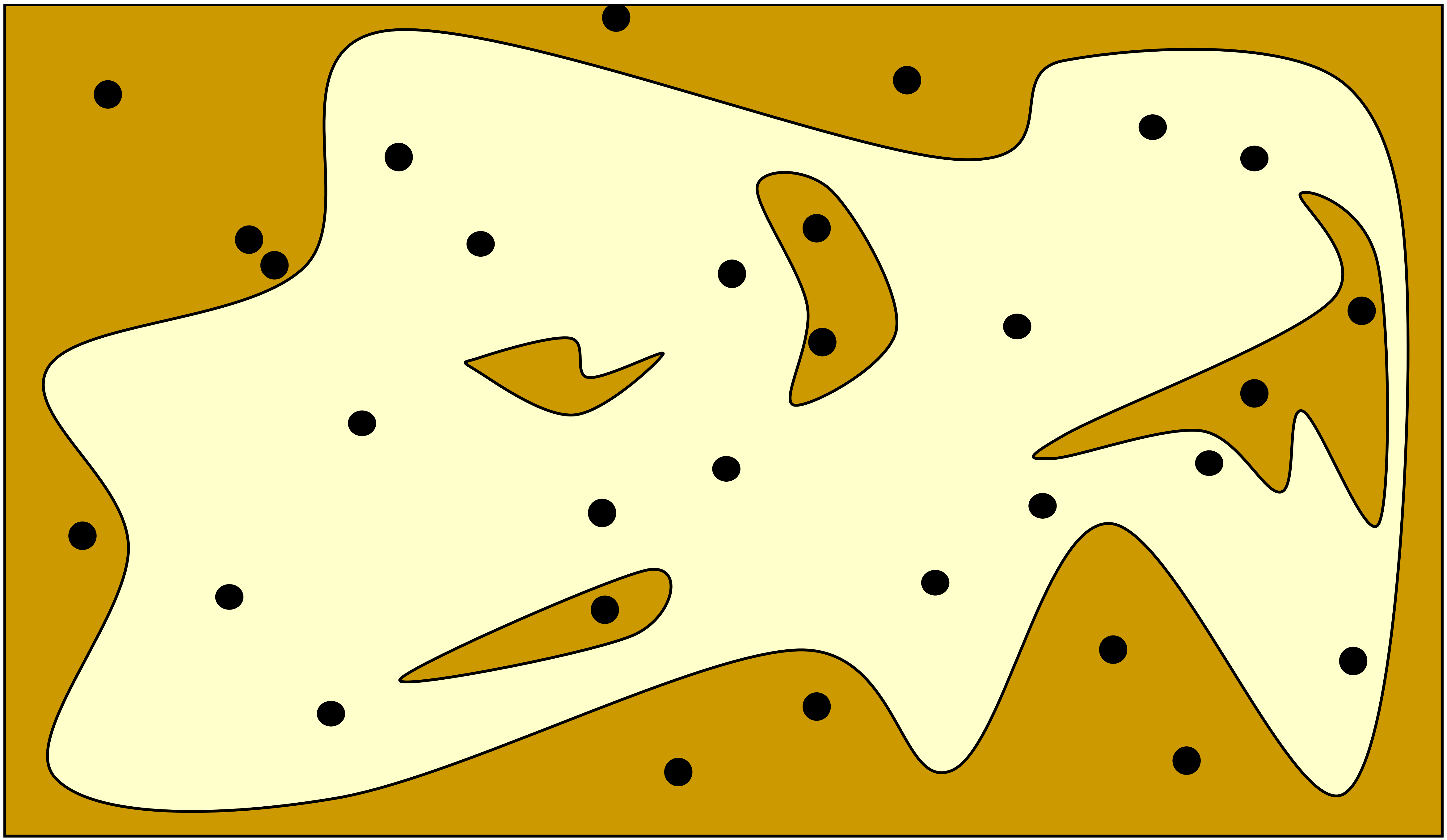
Randomly Sample Configurations



Slide from Pieter Abbeel.

Probabilistic Roadmaps

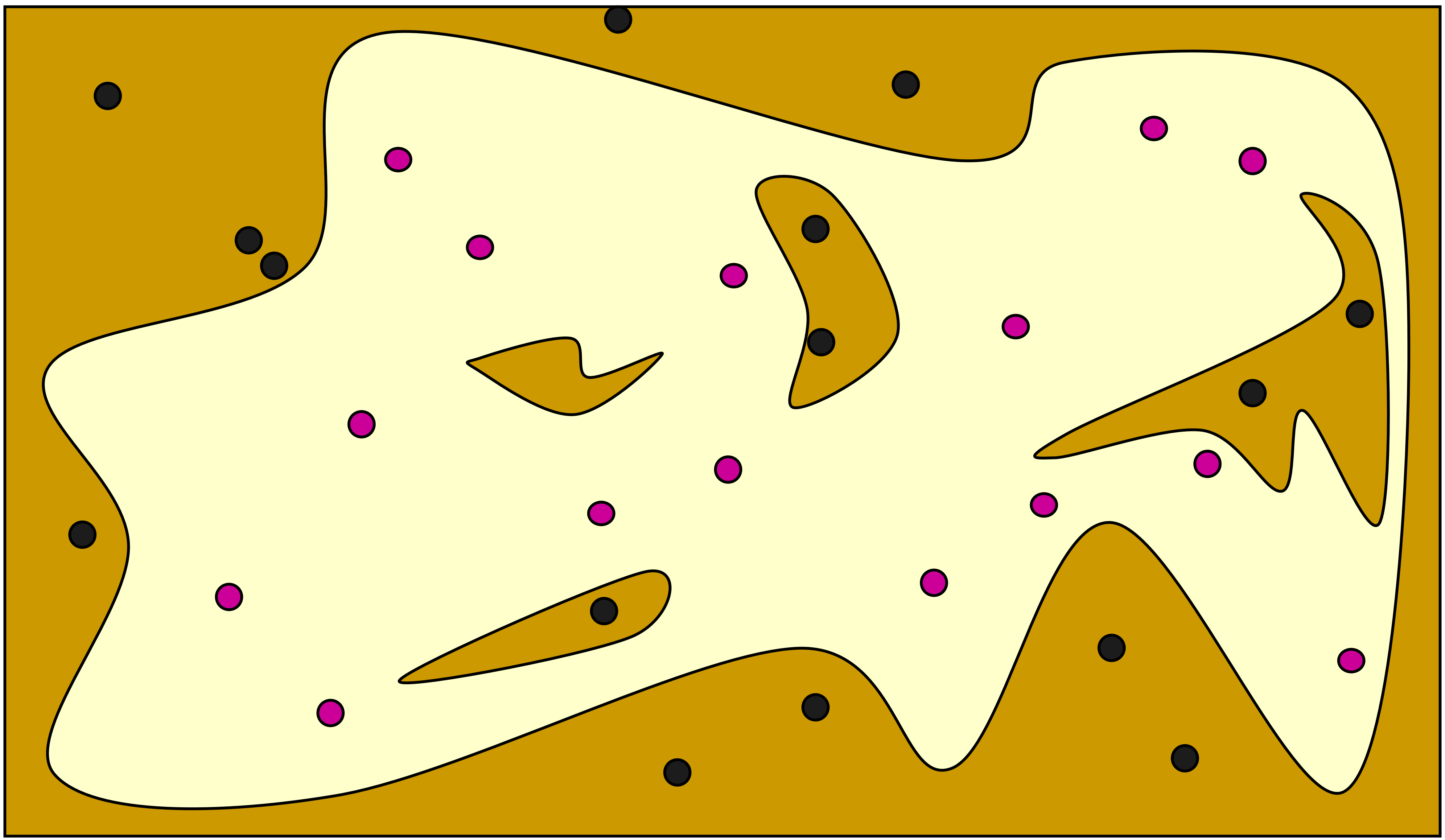
Randomly Sample Configurations



Slide from Pieter Abbeel.

Probabilistic Roadmaps

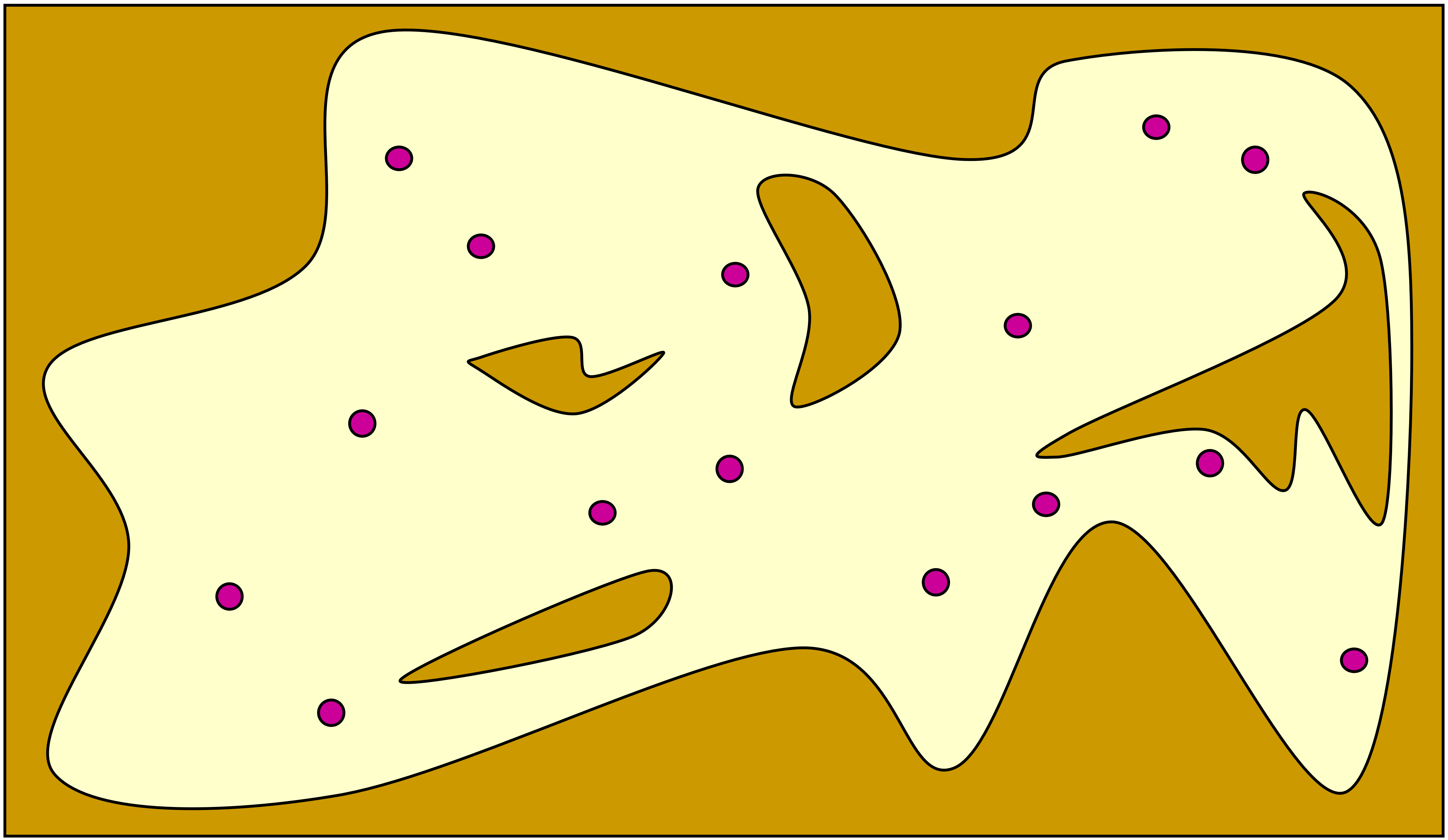
Test Sampled Configurations for Collisions



Slide from Pieter Abbeel.

Probabilistic Roadmaps

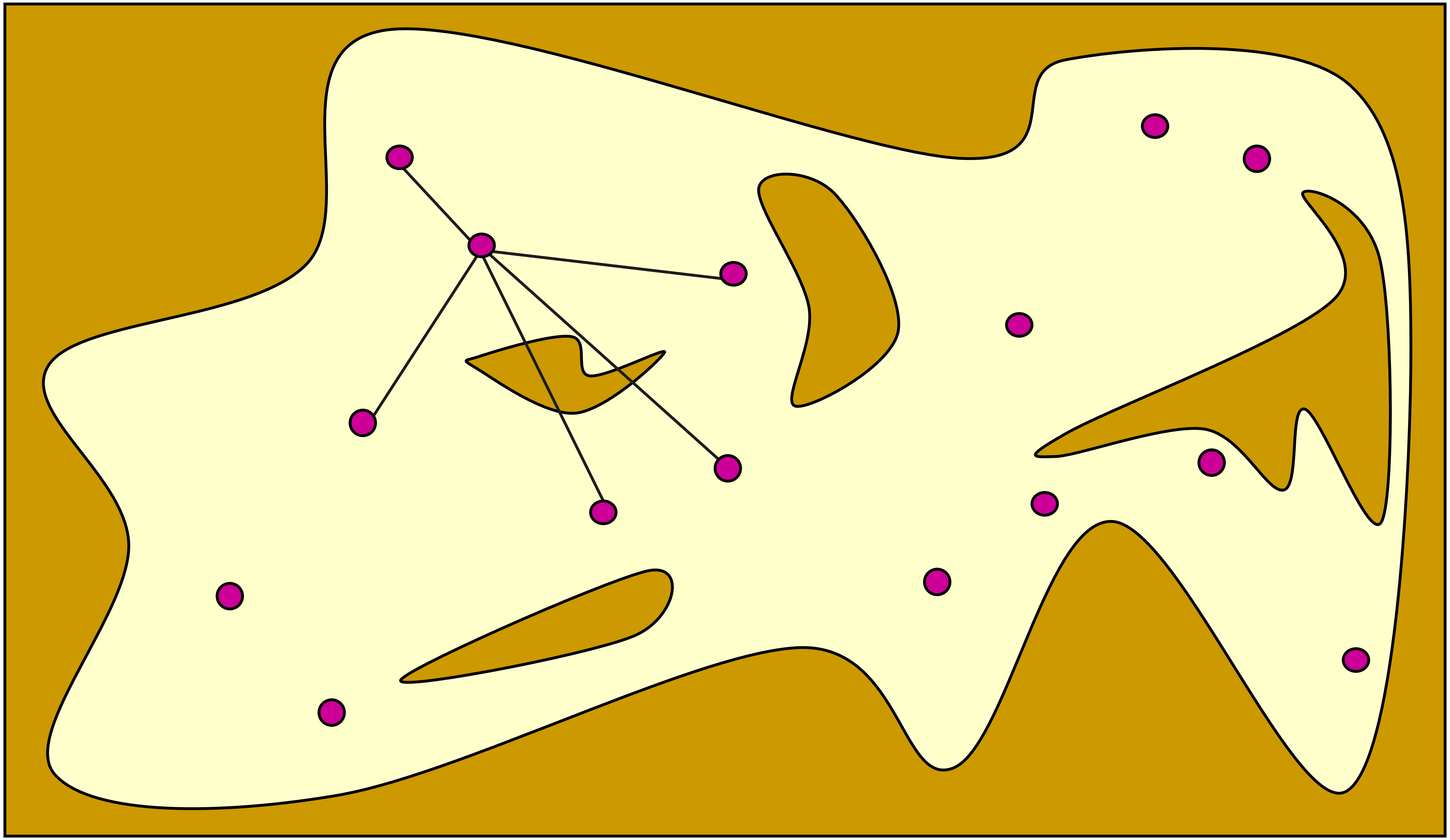
The collision-free configurations are retained as milestones



Slide from Pieter Abbeel.

Probabilistic Roadmaps

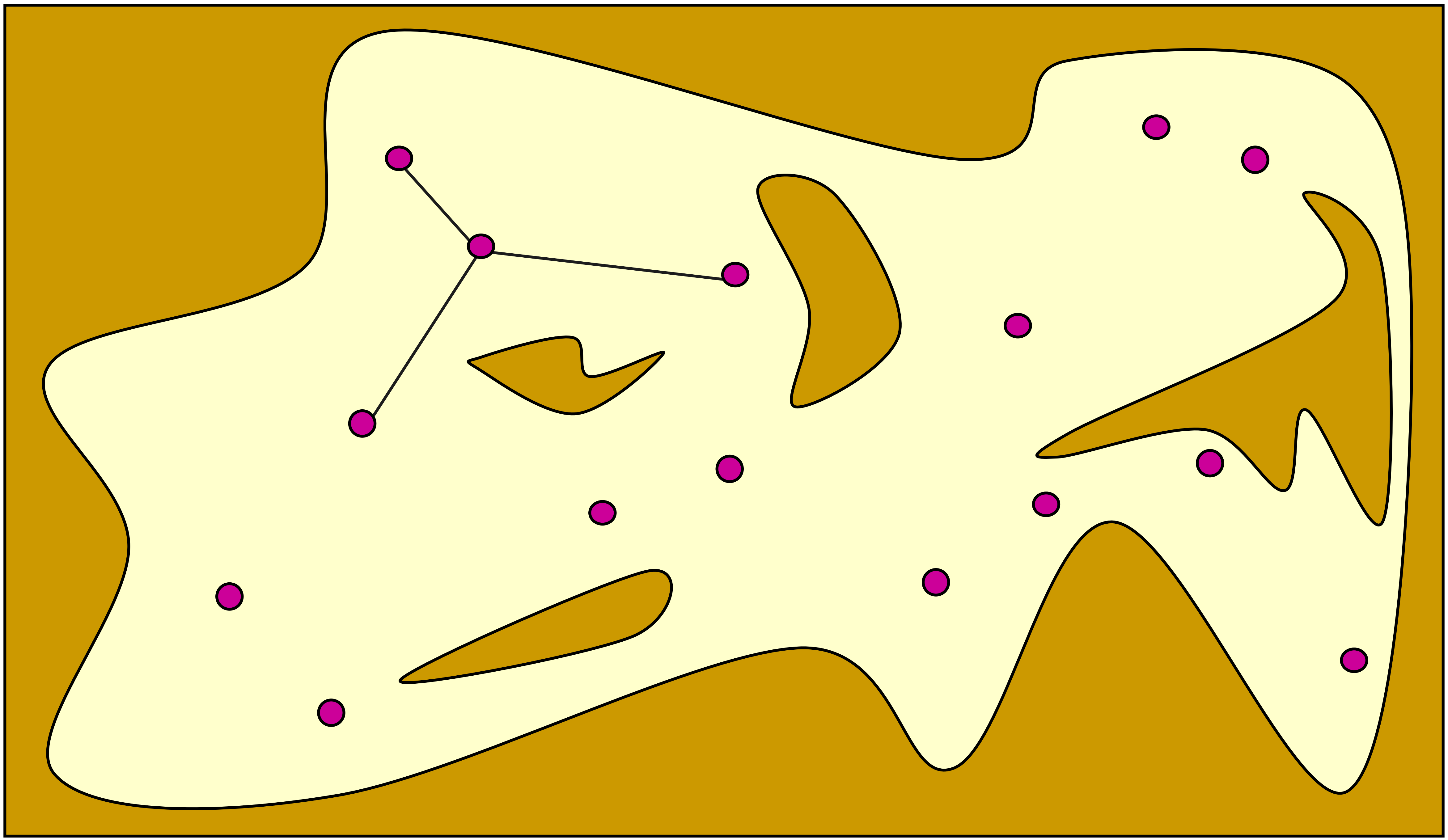
Each milestone is linked by straight paths to its nearest neighbors



Slide from Pieter Abbeel.

Probabilistic Roadmaps

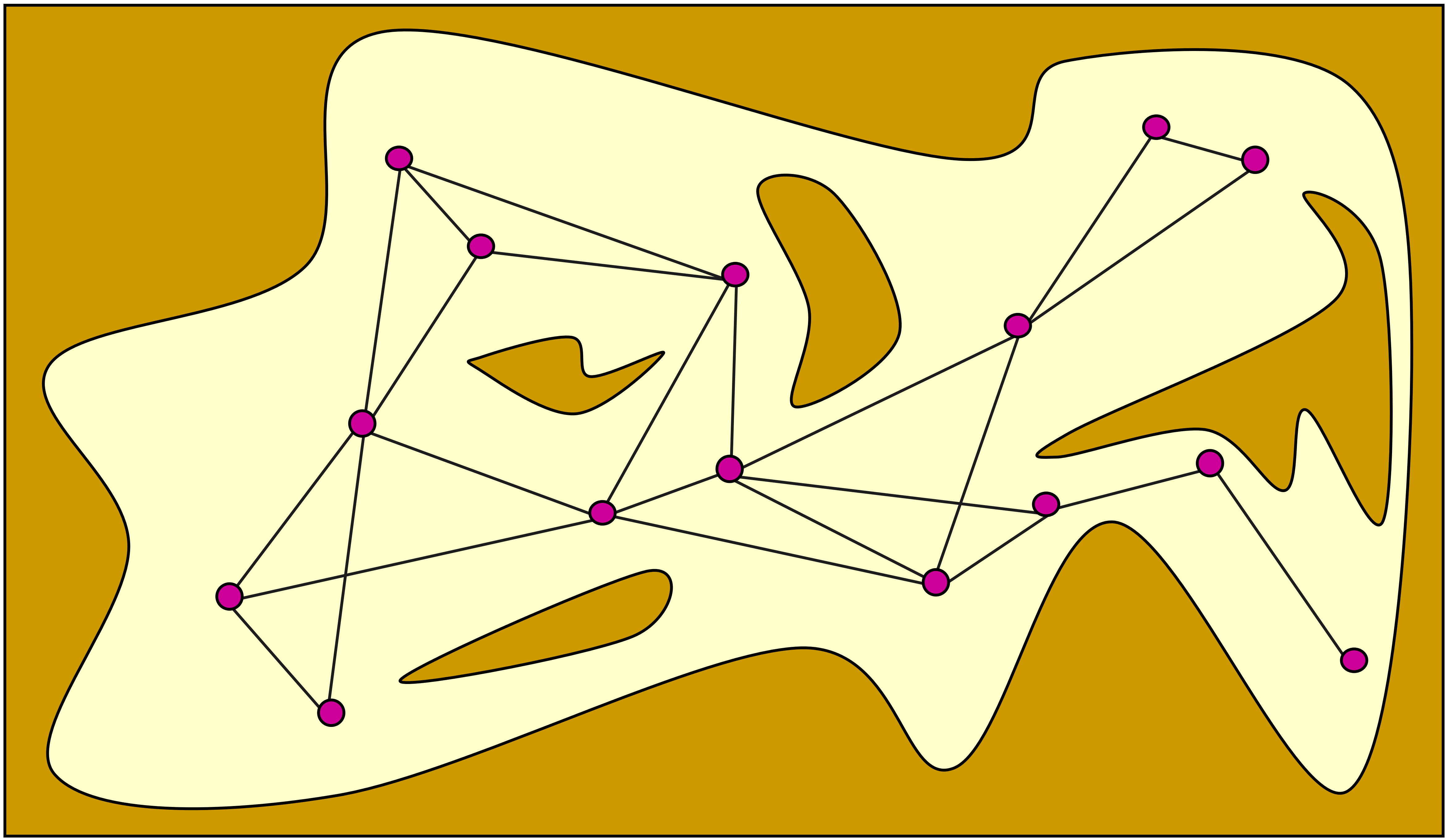
Paths that undergo collisions are removed



Slide from Pieter Abbeel.

Probabilistic Roadmaps

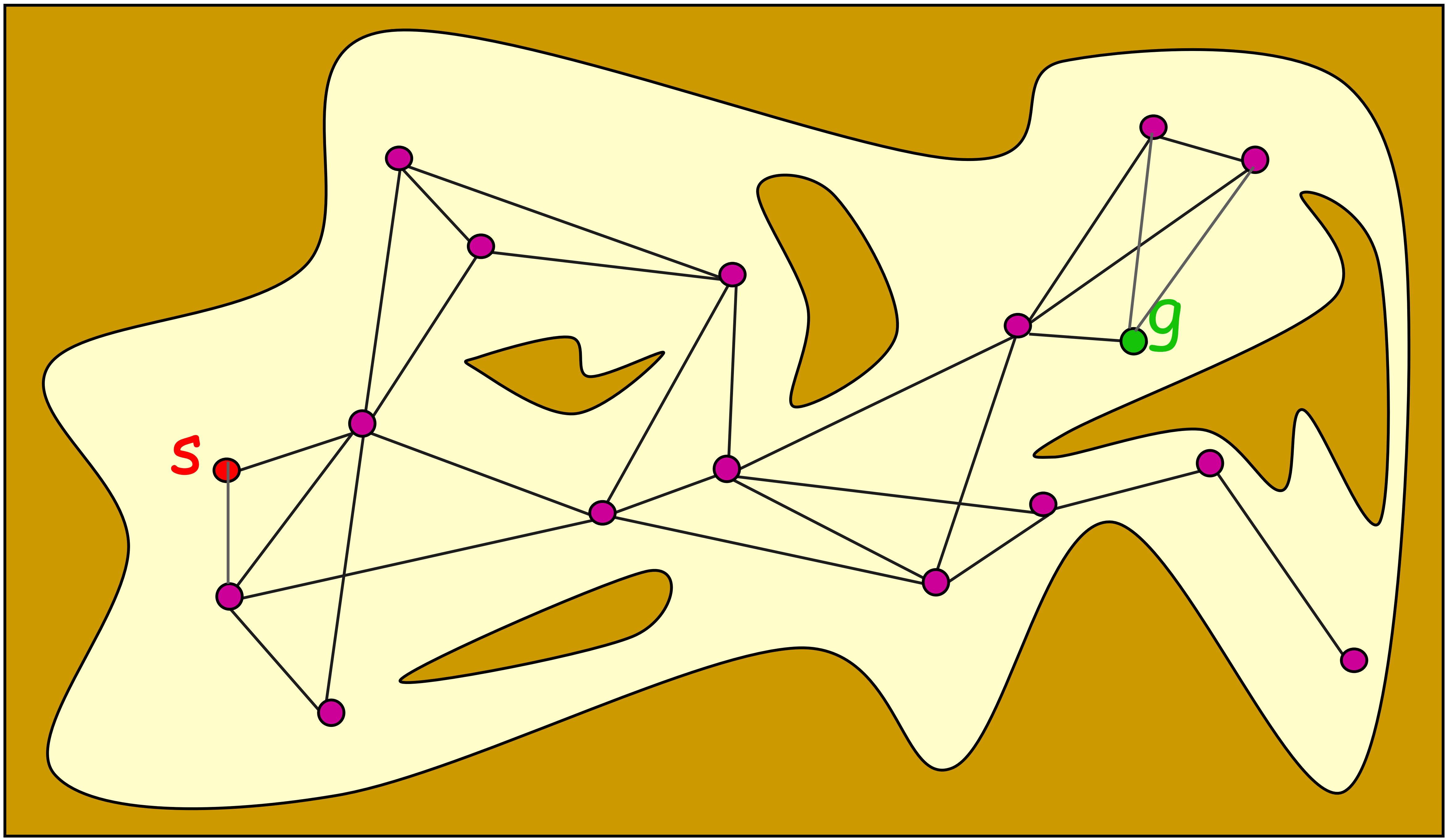
The collision-free links are retained as local paths to form the PRM



Slide from Pieter Abbeel.

Probabilistic Roadmaps

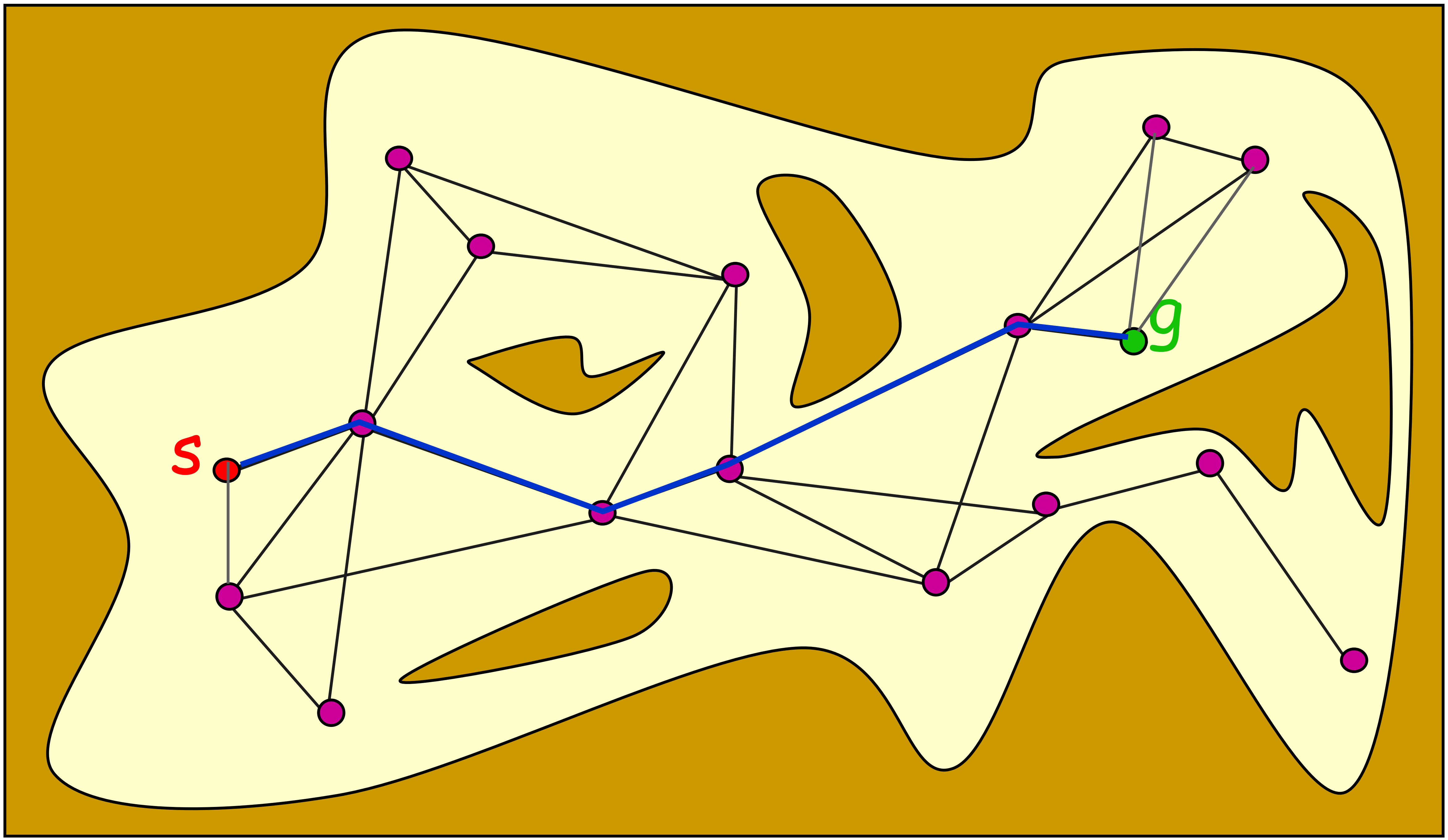
The start and goal configurations are included as milestones



Slide from Pieter Abbeel.

Probabilistic Roadmaps

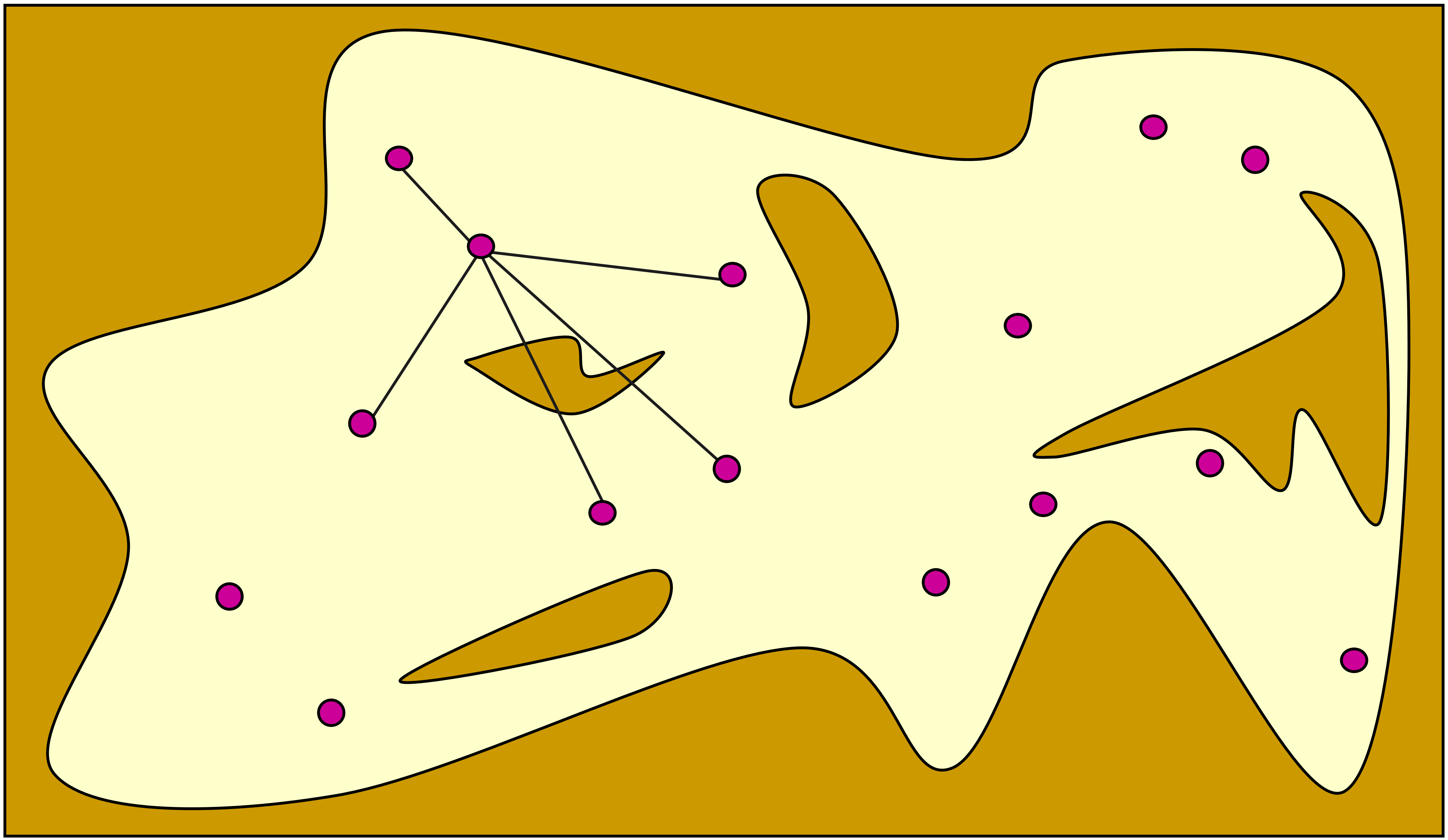
The PRM is searched for a path from s to g



Slide from Pieter Abbeel.

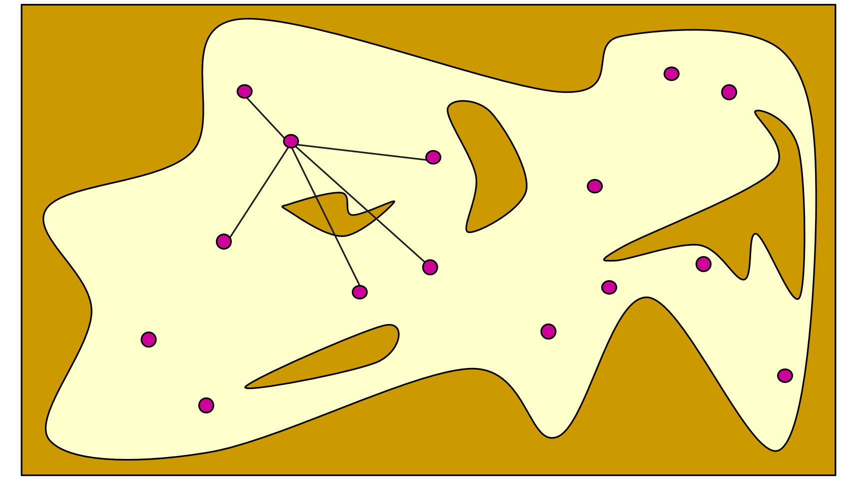
Probabilistic Roadmaps

Challenging to link milestones.



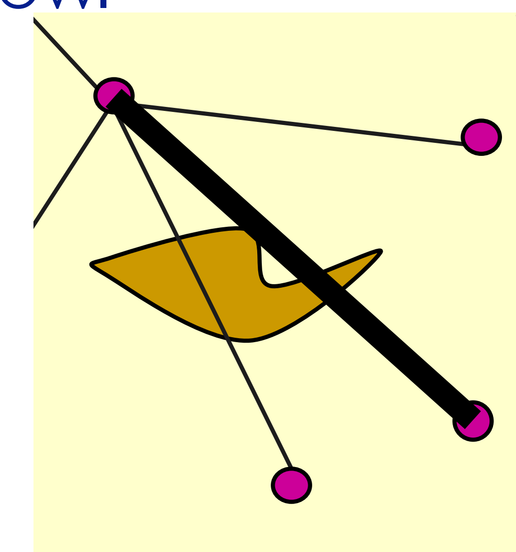
Slide from Pieter Abbeel.

Probabilistic Roadmaps

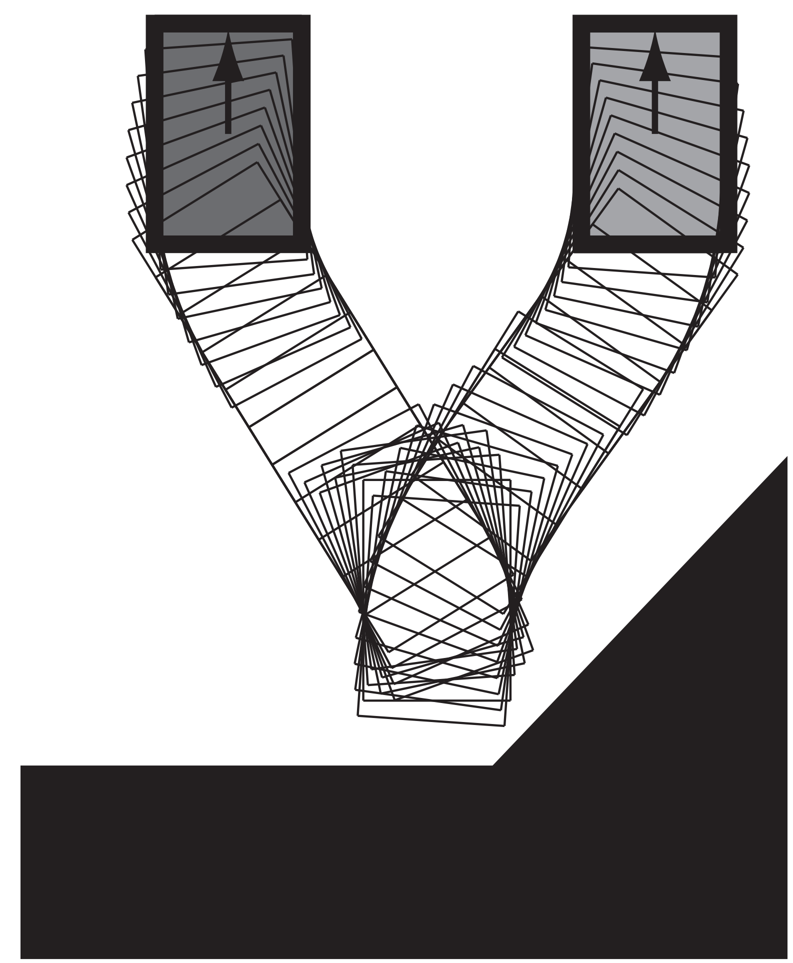


Challenging to link milestones.

Collision checking can be slow.



All straight line paths may not be feasible, or a good measure of distance between states.

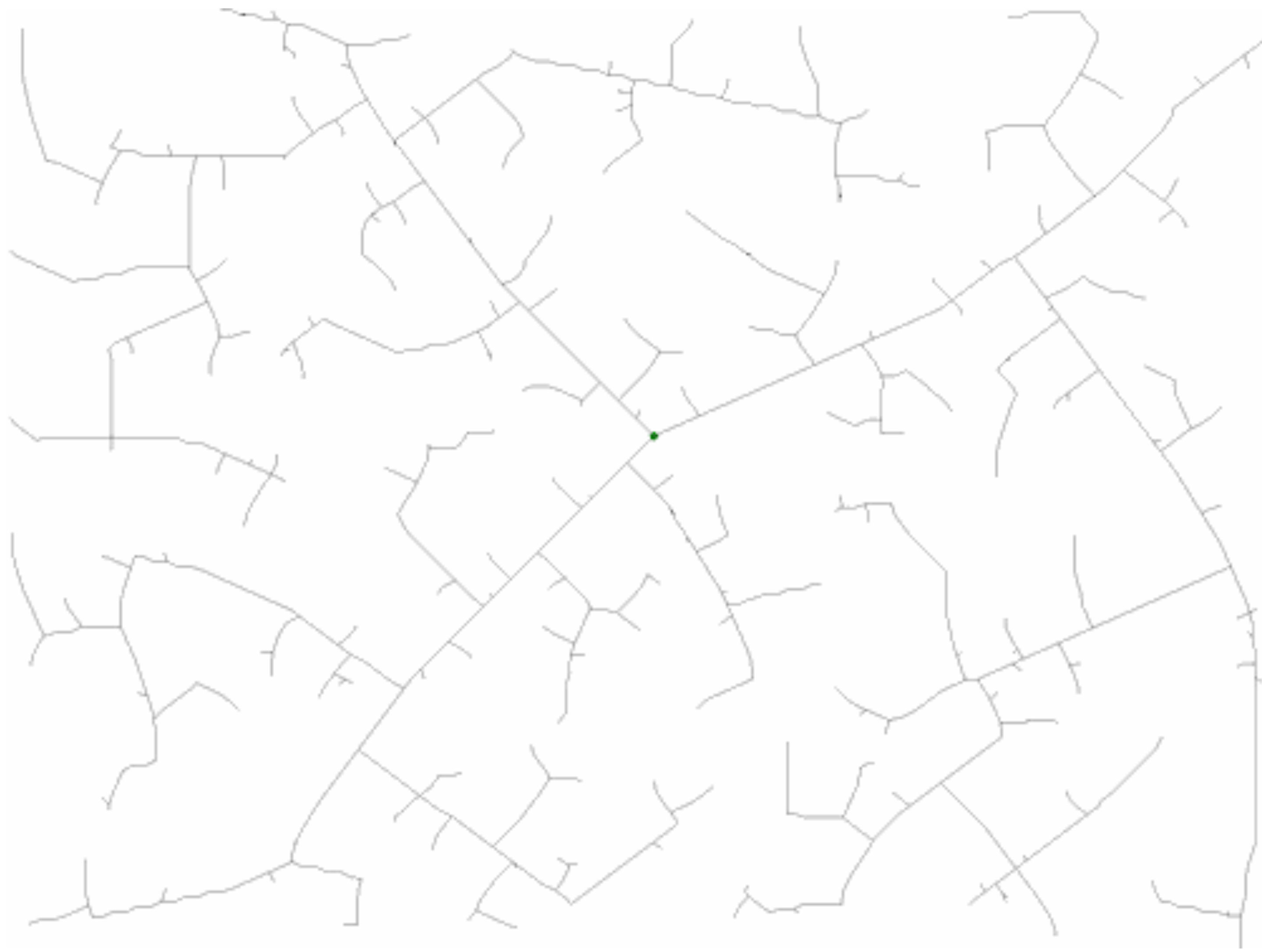


Slide from Pieter Abbeel, Modern Robotics.

Rapidly Exploring Random Trees (RRTs)

Kinodynamic planning

Build up a tree through generating "next states" in the tree by executing random controls.



Slide from Pieter Abbeel.

Rapidly Exploring Random Trees (RRTs)

Build up a tree through generating "next states" in the tree by executing random controls.

GENERATE_RRT($x_{init}, K, \Delta t$)

$\mathcal{T}$.init(x_{init});
for $k = 1$ **to** K **do**
  $x_{rand} \leftarrow$ RANDOM_STATE();
  $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$);
  $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
  $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
  $\mathcal{T}$.add_vertex(x_{new});
  $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);
Return $\mathcal{T}$

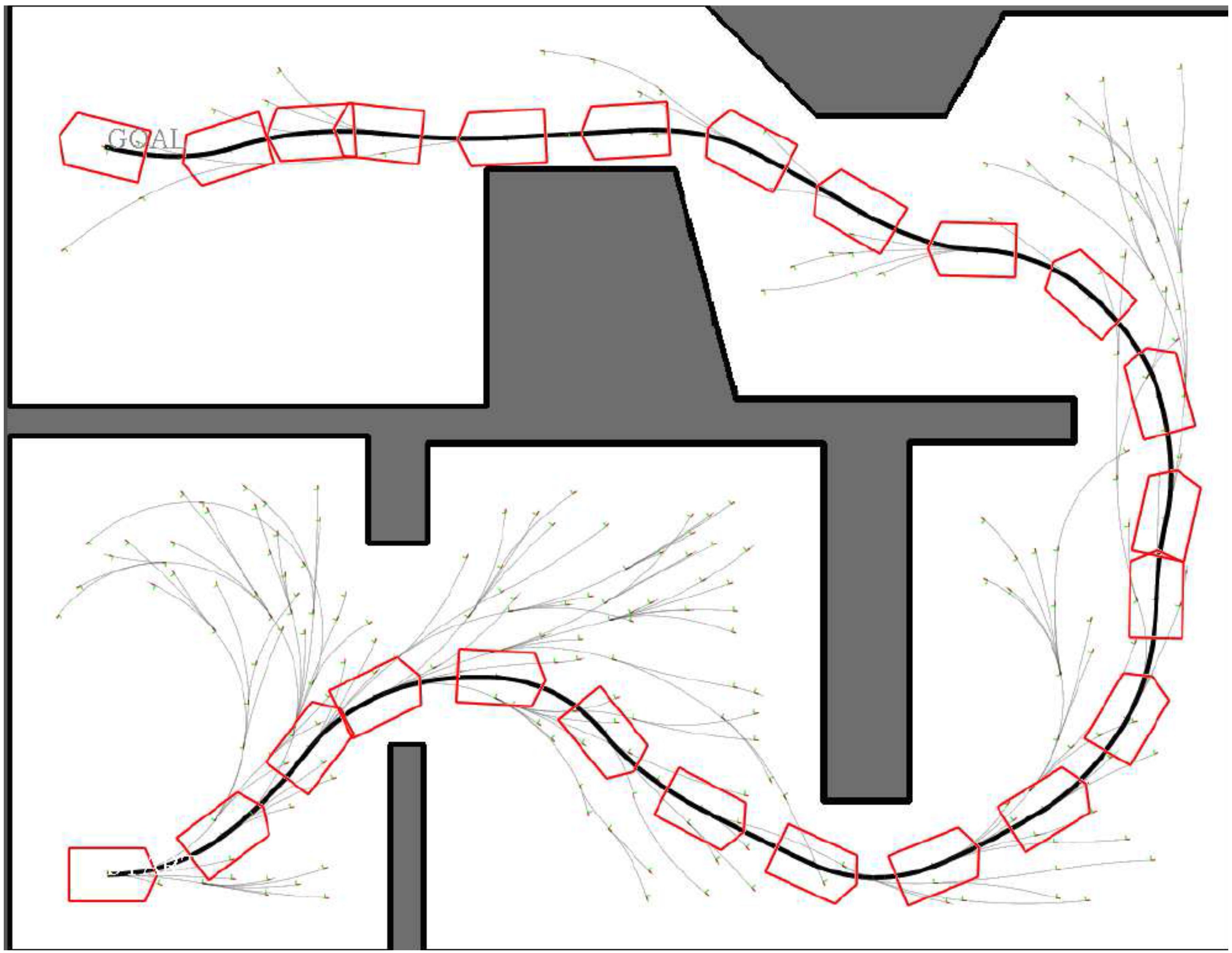
SELECT_INPUT(x_{rand}, x_{near})

- Two point boundary value problem
  - If too hard to solve, often just select best out of a set of control sequences. This set could be random, or some well chosen set of primitives.

Slide from Pieter Abbeel.

Rapidly Exploring Random Trees (RRTs)

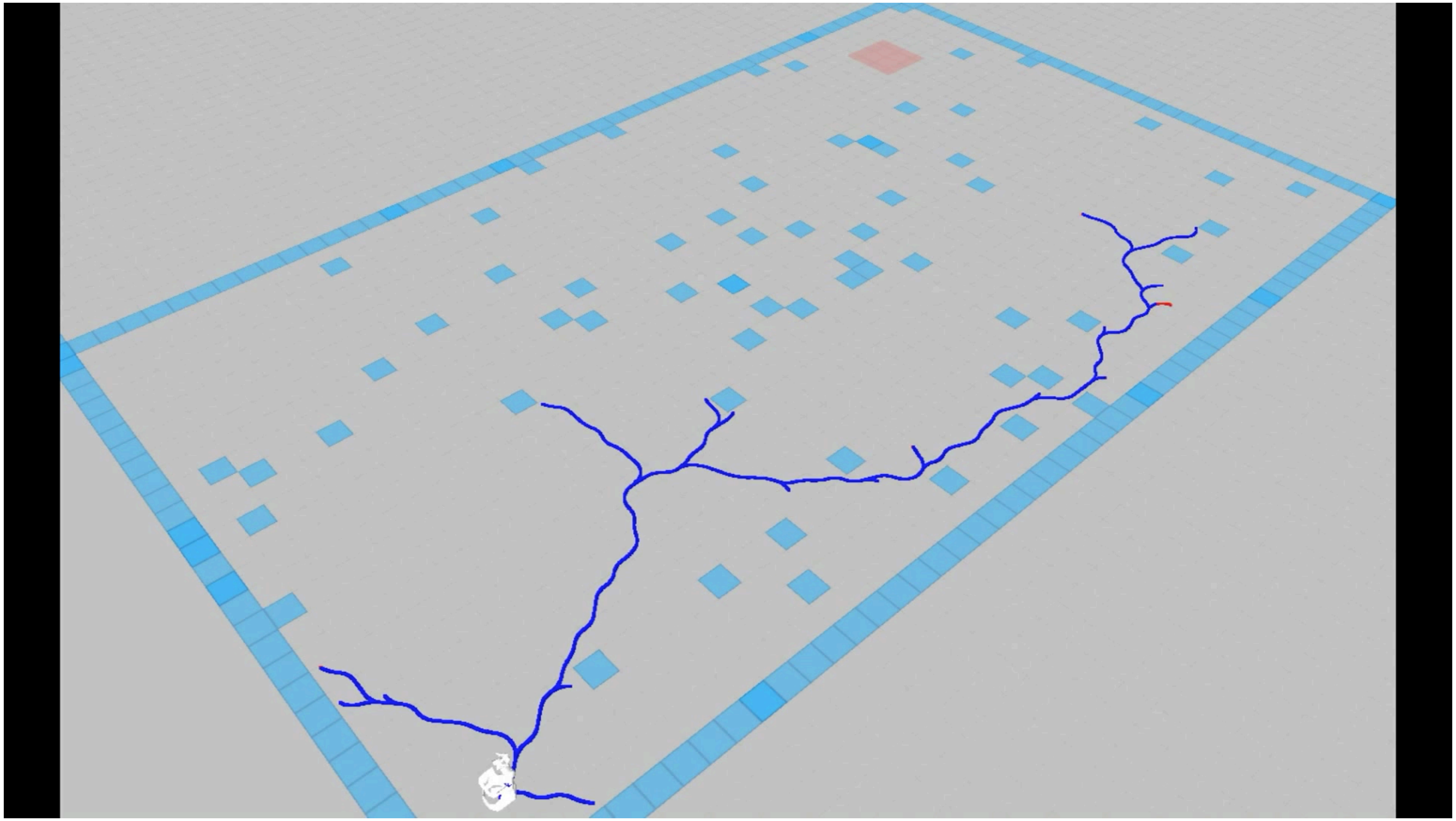
Build up a tree through generating "next states" in the tree by executing random controls.



Slide from Pieter Abbeel.

Rapidly Exploring Random Trees (RRTs)

Build up a tree through generating "next states" in the tree by executing random controls.



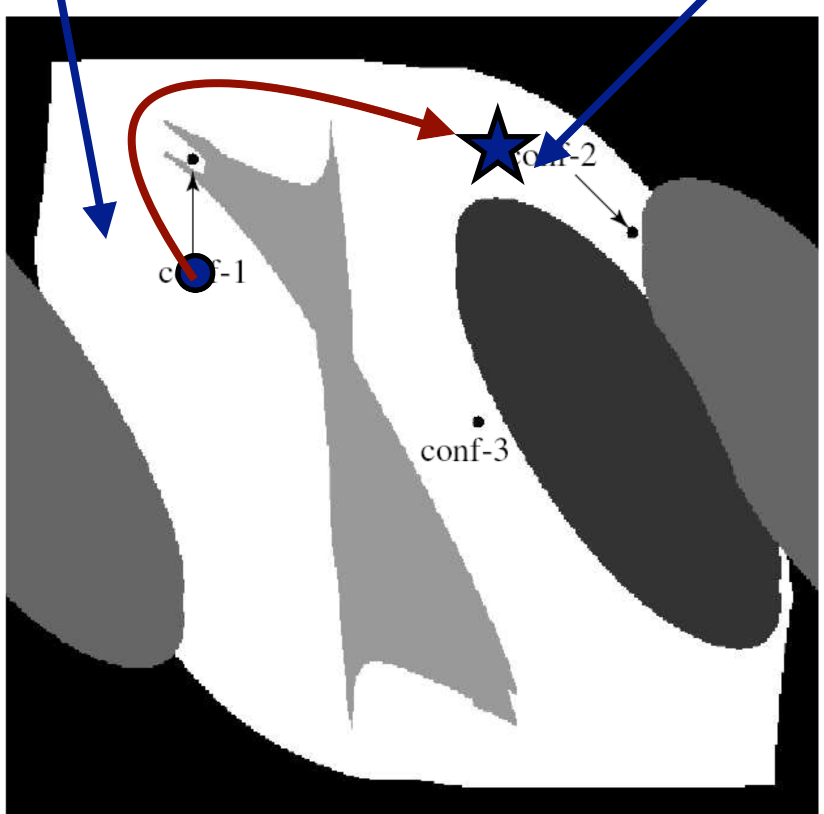
How to move your robot?

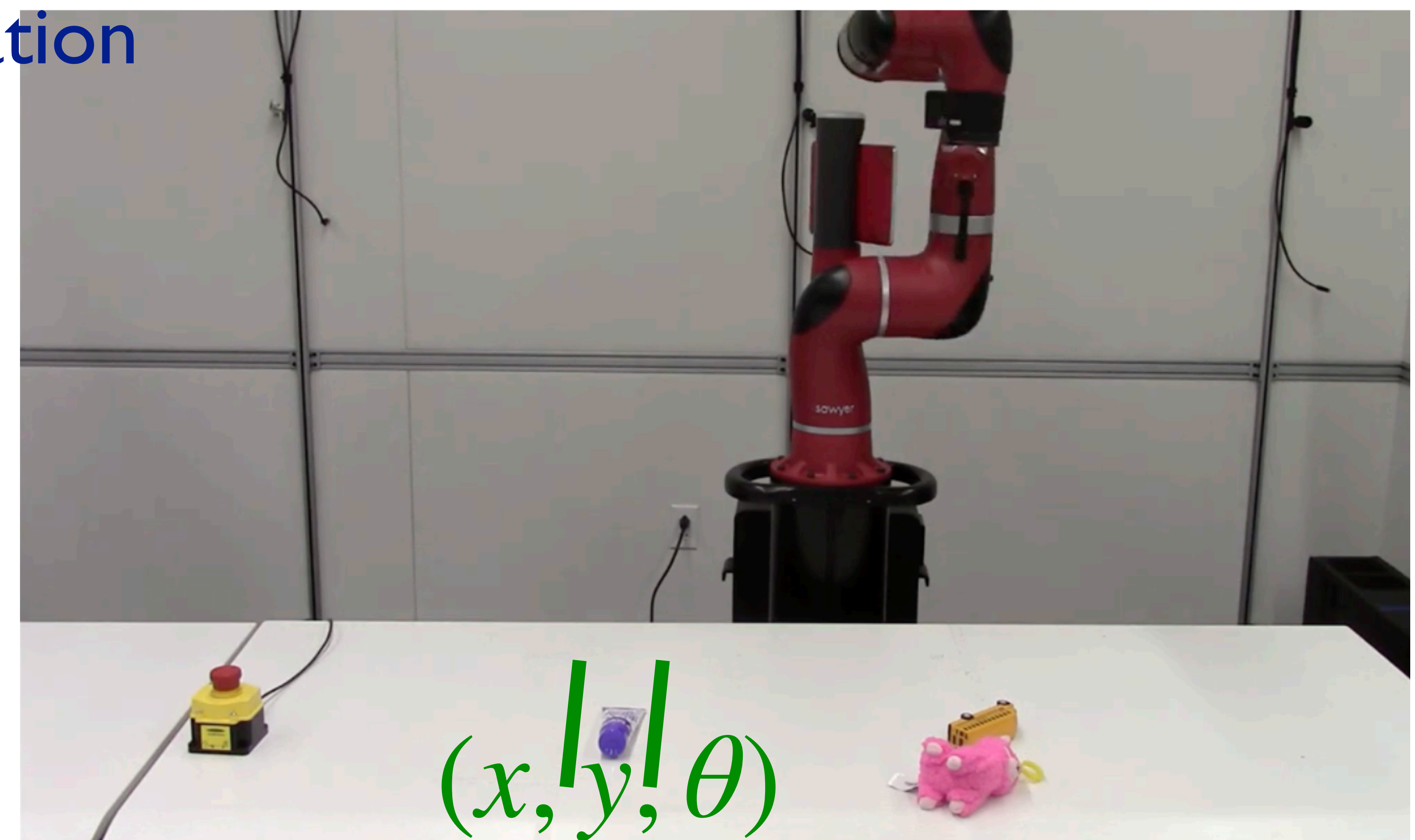
1. Task space to Configuration space

2. Configuration space trajectory

Initial configuration

Desired configuration





How to move your robot?

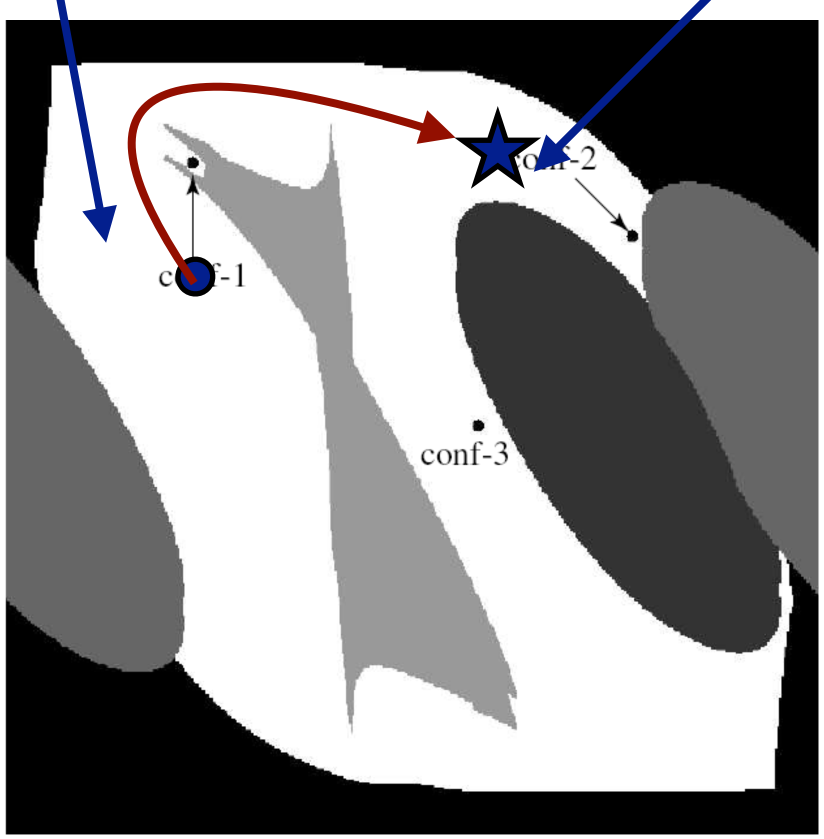
1. Task space to Configuration space

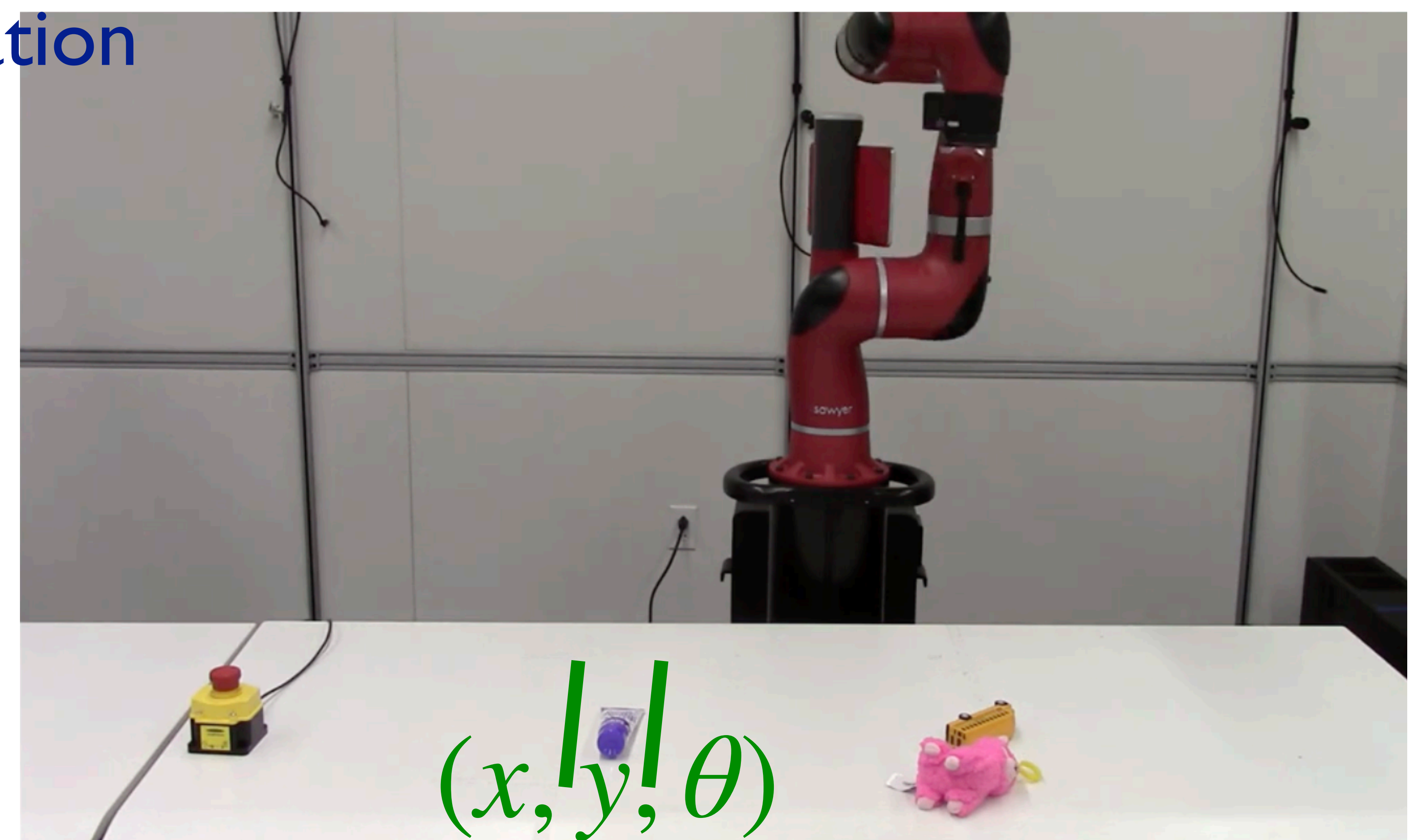
2. Configuration space trajectory

3. Trajectory execution

Initial configuration

Desired configuration





Trajectory Execution

Dynamically feasible trajectory x_t^{ref} from planner

What control commands should I apply in order to get the robot to robustly track this trajectory?

Robot state	x_t	Robot location, or joint angles.
Control sequence	u_t	Velocities, torques.
Dynamics function	$x_{t+1} = f(x_t, u_t)$	State evolution as we apply control.
Cost function	$\sum_t \lVert x_t - x_t^{ref} \rVert$	

Low-level control can be formulated as an optimization problem.

Trajectory Execution

Feedback Control

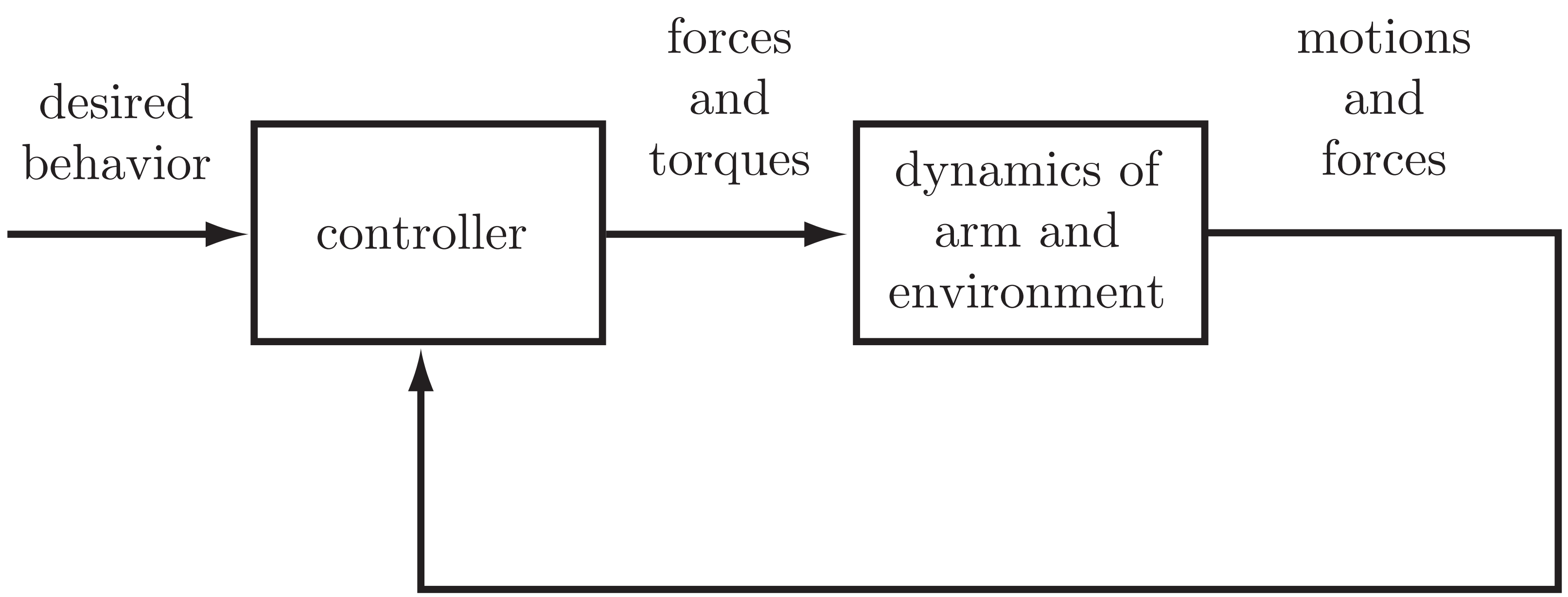


Figure from Modern Robotics.

Low-level Control

Simplifying assumptions:
Linear dynamics, quadratic cost.

$\Rightarrow$

Linear Quadratic Regulator

Exactly solved using dynamic programming.

The LQR setting assumes a linear dynamical system:

$$x_{t+1} = Ax_t + Bu_t,$$

x_t: state at time t
u_t: input at time t
It assumes a quadratic cost function:

$$g(x_t, u_t) = x_t^\top Q x_t + u_t^\top R u_t$$

with $Q \succ 0, R \succ 0$.

For a square matrix X we have $X \succ 0$ if and only if for all vectors z we have $z^\top X z > 0$. Hence there is a non-zero cost for any state different from the all-zeros state, and any input different from the all-zeros input.

Slide from Pieter Abbeel.

Linear Quadratic Regulator

Cost if the system is in state x,
and we have *i steps to go.*

$$J_i(x)$$

Cost if the system is in state x,
and we have *i+1 steps to go.*

$$J_{i+1}(x)$$

$$= min_u \; x^T Q x + u^T R u + J_i(Ax + Bu)$$

Linear Quadratic Regulator

$$J_{i+1}(x) \;\leftarrow\; \min_u \left[x^\top Q x + u^\top R u + J_i(Ax+Bu)\right]$$

Initialize $J_0(x) = x^\top P_0 x$.

$$\begin{aligned} J_1(x) &= \min_u \left[x^\top Q x + u^\top R u + J_0(Ax+Bu)\right] \\ &= \min_u \left[x^\top Q x + u^\top R u + (Ax+Bu)^\top P_0 (Ax+Bu)\right] \qquad (1) \end{aligned}$$

To find the minimum over u, we set the gradient w.r.t. u equal to zero:

$$\nabla_u [\ldots] = 2Ru + 2B^\top P_0 (Ax+Bu) = 0,$$

$$\text{hence: } u = -(R + B^\top P_0 B)^{-1} B^\top P_0 A x \qquad (2)$$

$$\begin{aligned} \text{(2) into (1): } J_1(x) &= x^\top P_1 x \\ \text{for: } P_1 &= Q + K_1^\top R K_1 + (A + BK_1)^\top P_0 (A + BK_1) \\ K_1 &= -(R + B^\top P_0 B)^{-1} B^\top P_0 A. \end{aligned}$$

Slide from Pieter Abbeel.

Linear Quadratic Regulator

In summary:

$$J_0(x) = x^\top P_0 x$$
$$x_{t+1} = Ax_t + Bu_t$$
$$g(x,u) = u^\top R u + x^\top Q x$$

$$\begin{aligned} J_1(x) &= x^\top P_1 x \\ \text{for: } P_1 &= Q + K_1^\top R K_1 + (A + BK_1)^\top P_0 (A + BK_1) \\ K_1 &= -(R + B^\top P_0 B)^{-1} B^\top P_0 A. \end{aligned}$$

$J_1(x)$ is quadratic, just like $J_0(x)$.

Update is the same for all times and can be done in closed form for this particular continuous state-space system and cost!

$$\begin{aligned} J_2(x) &= x^\top P_2 x \\ \text{for: } P_2 &= Q + K_2^\top R K_2 + (A + BK_2)^\top P_1 (A + BK_2) \\ K_2 &= -(R + B^\top P_1 B)^{-1} B^\top P_1 A. \end{aligned}$$

Slide from Pieter Abbeel.

Linear Quadratic Regulator

Set $P_0 = 0$.
for $i = 1, 2, 3, \ldots$

$$\begin{aligned} K_i &= -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A \\ P_i &= Q + K_i^\top R K_i + (A + B K_i)^\top P_{i-1} (A + B K_i) \end{aligned}$$

The optimal policy for a i-step horizon is given by:

$$\pi(x) = K_i x$$

The cost-to-go function for a i-step horizon is given by:

$$J_i(x) = x^\top P_i x.$$

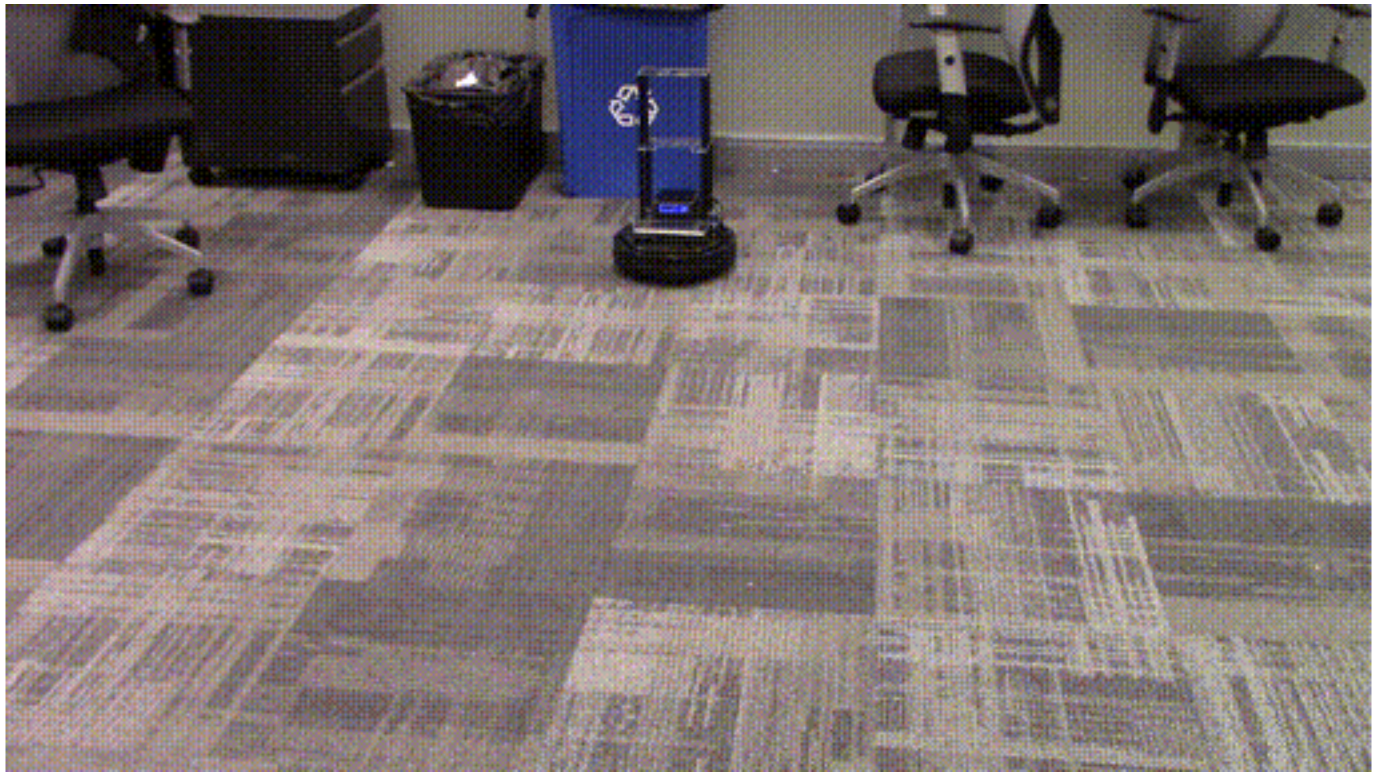
Slide from Pieter Abbeel.

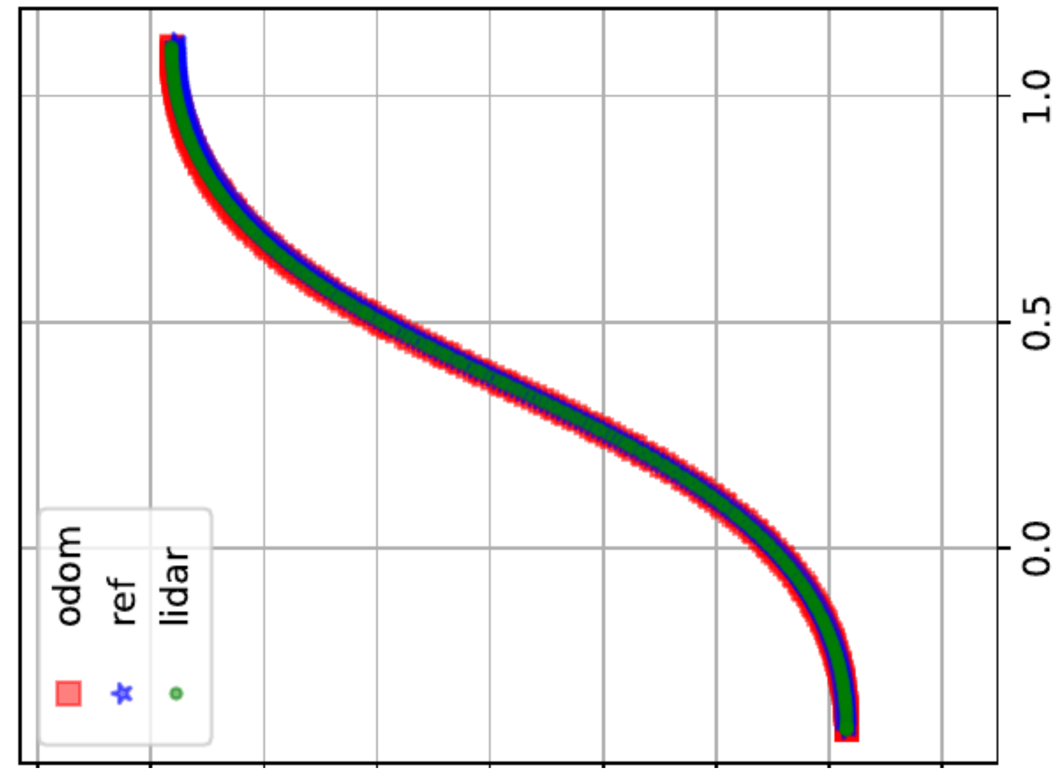
Linear Quadratic Regulator

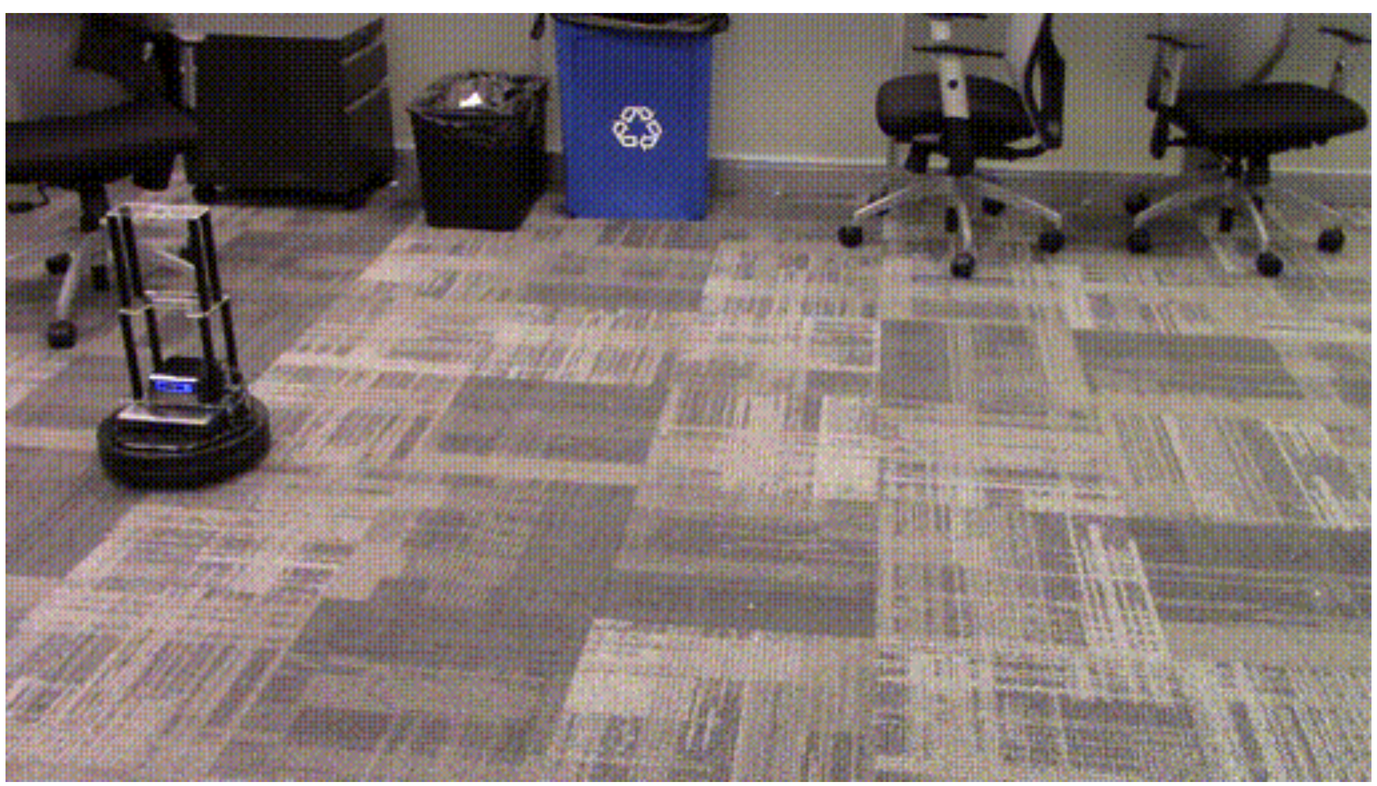
Extensions which make it more generally applicable:

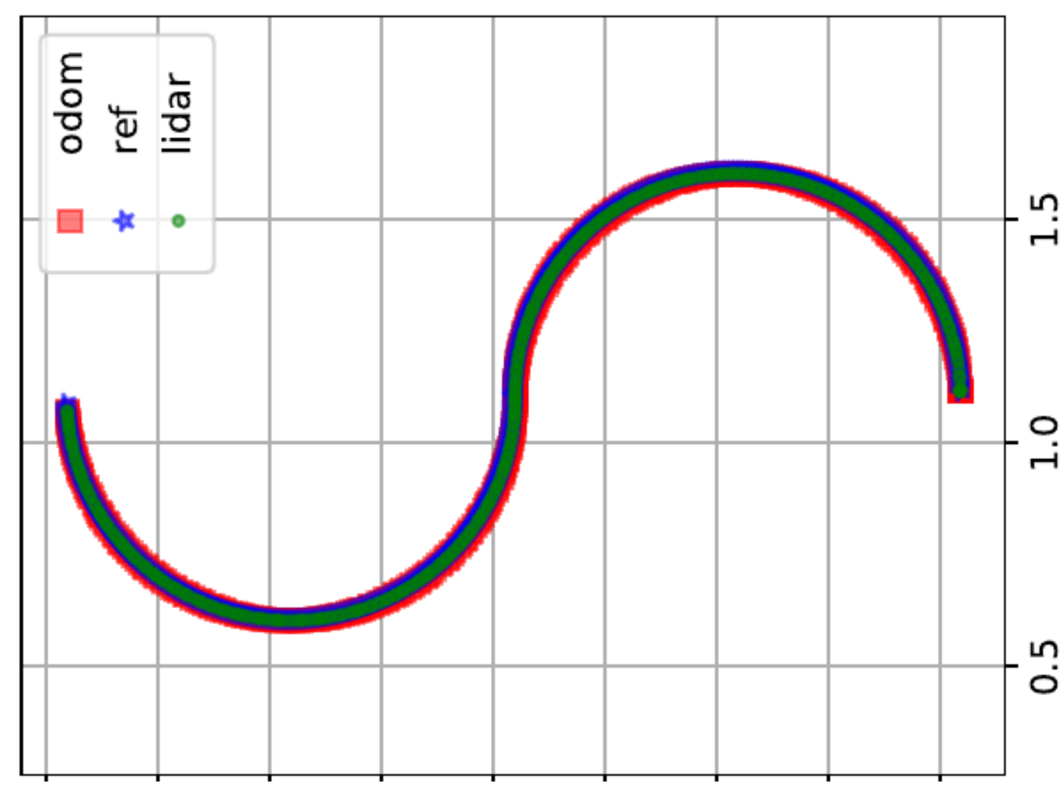
- Affine systems System with stochasticity
- Regulation around non-zero fixed point for non-linear systems
- Penalization for change in control inputs
- Linear time varying (LTV) systems
- Trajectory following for non-linear systems

Linear Quadratic Regulator









How to move your robot?

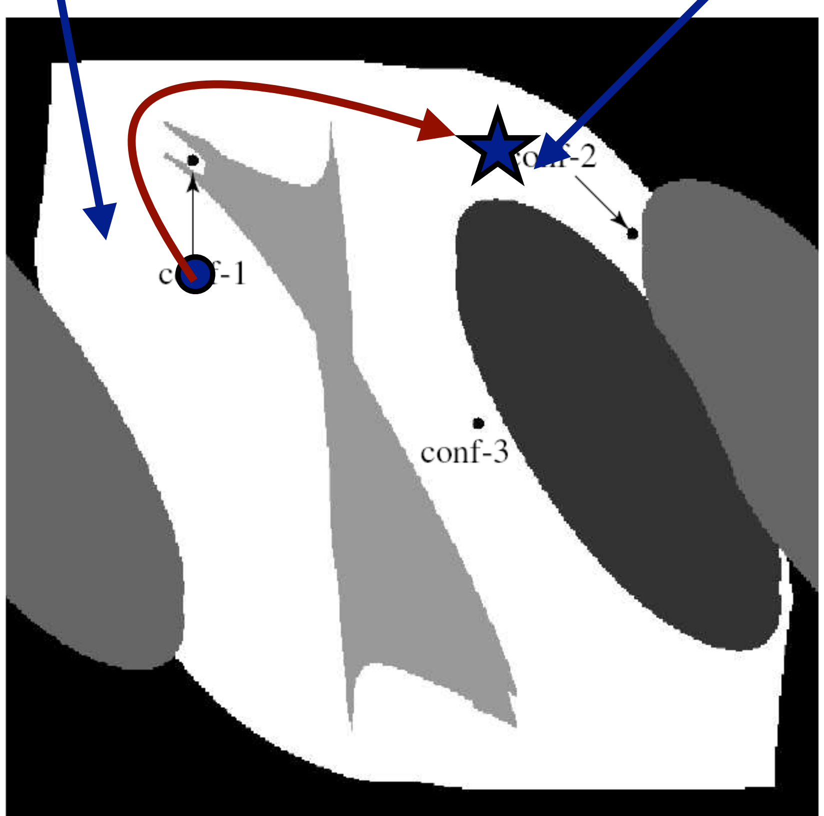
1. Task space to Configuration space

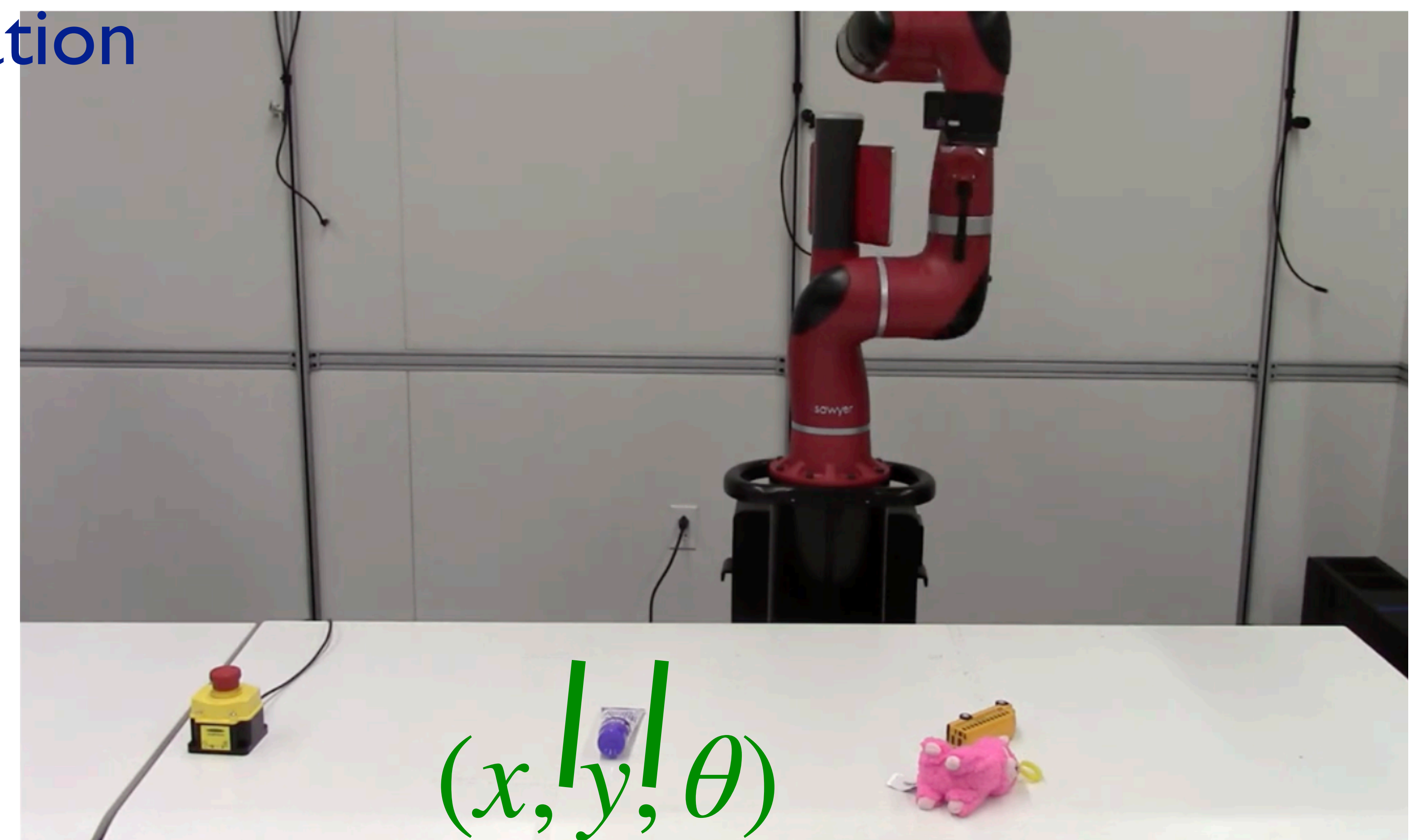
2. Configuration space trajectory

3. Trajectory tracking

Initial configuration

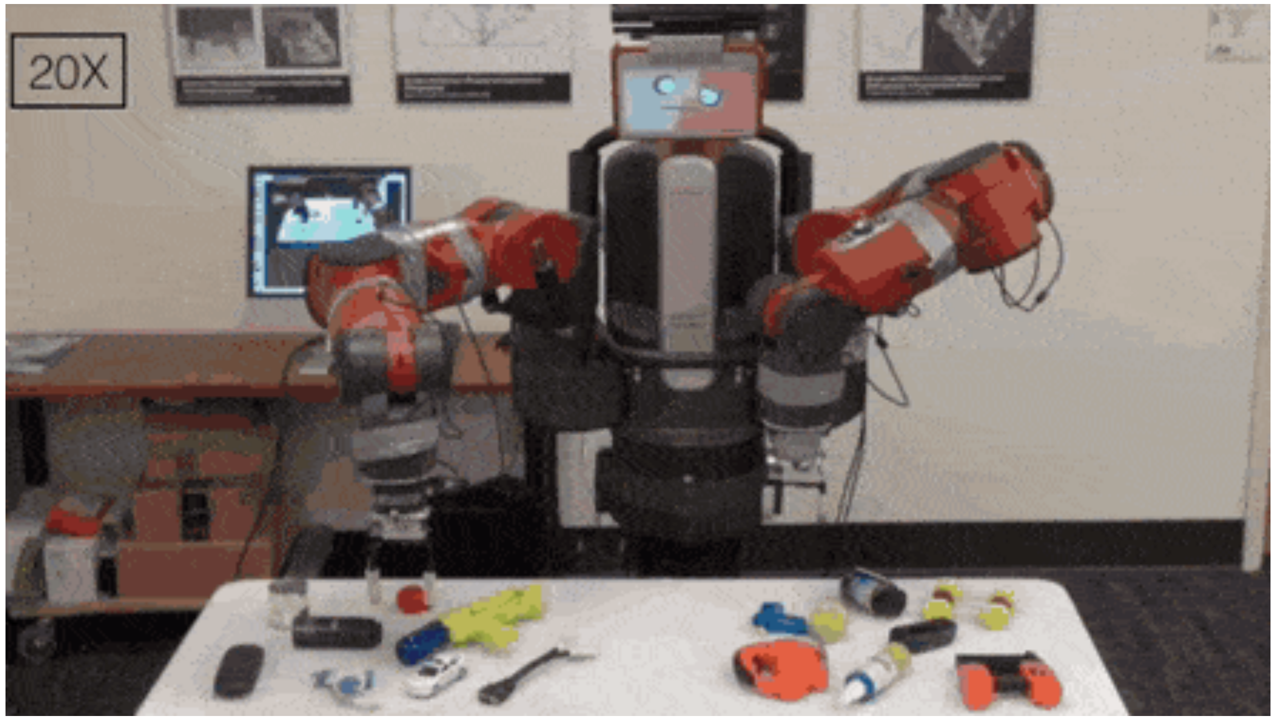
Desired configuration





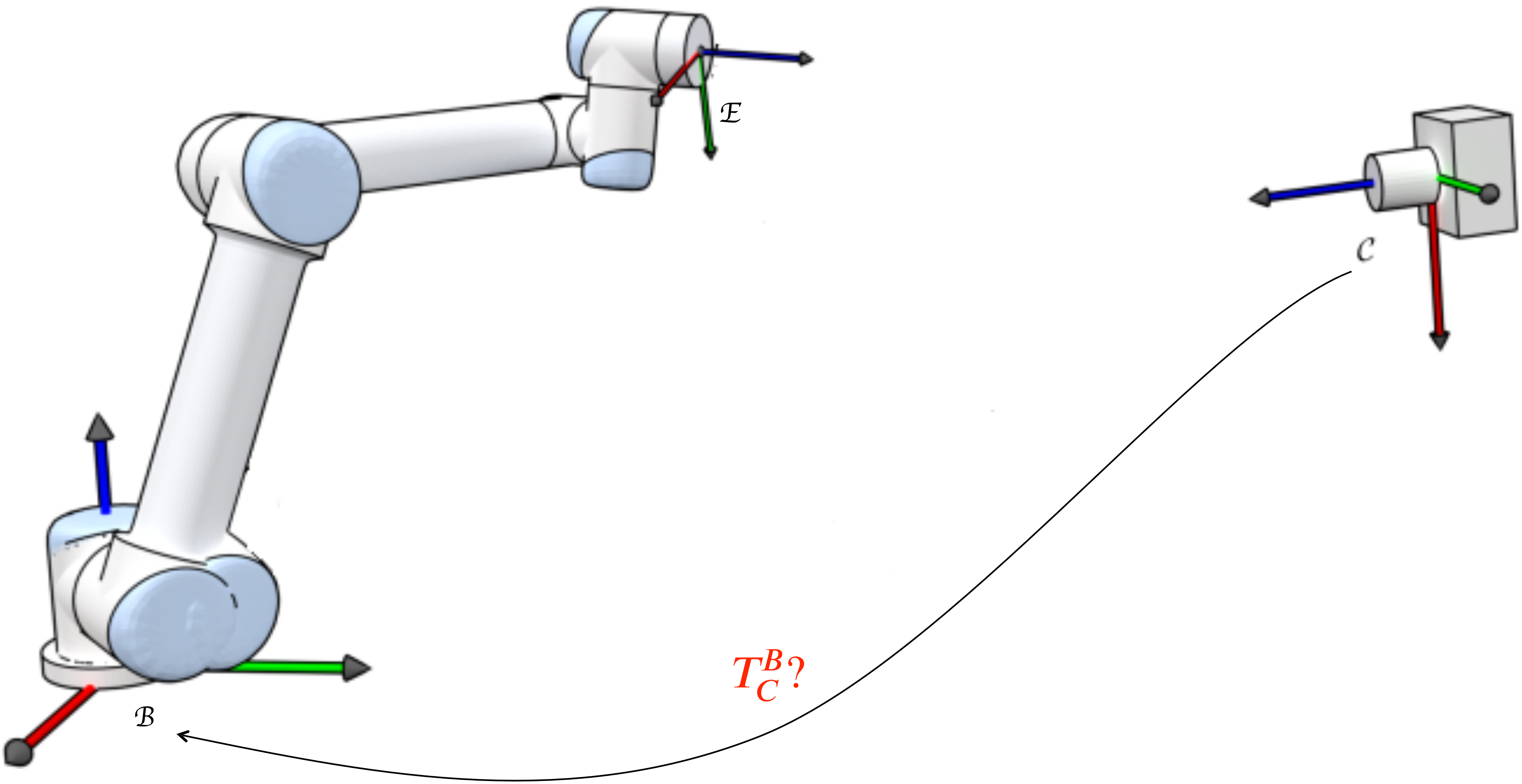
Minor Detail

Camera Calibration



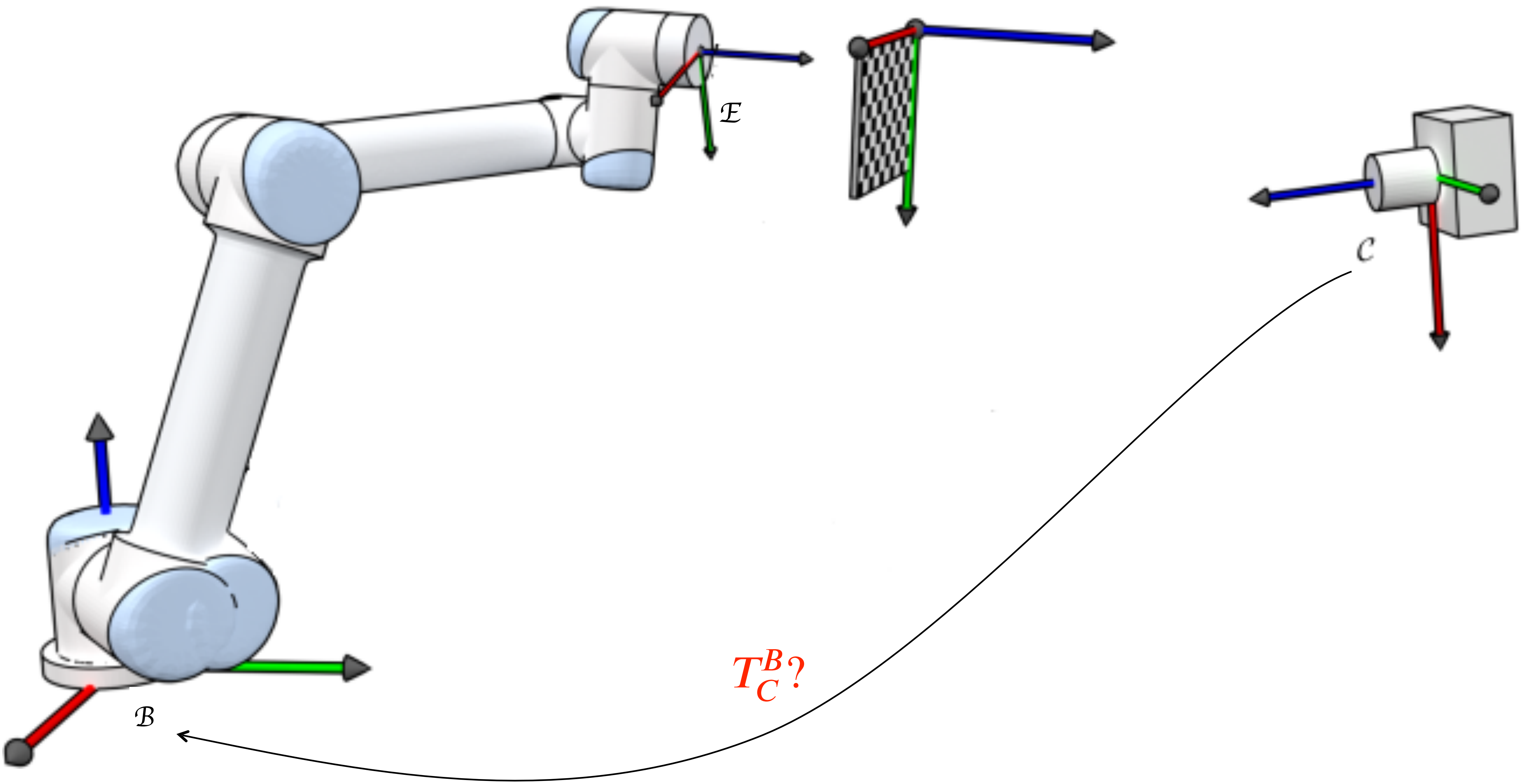
Images from Lerrel Pinto.

Camera Calibration



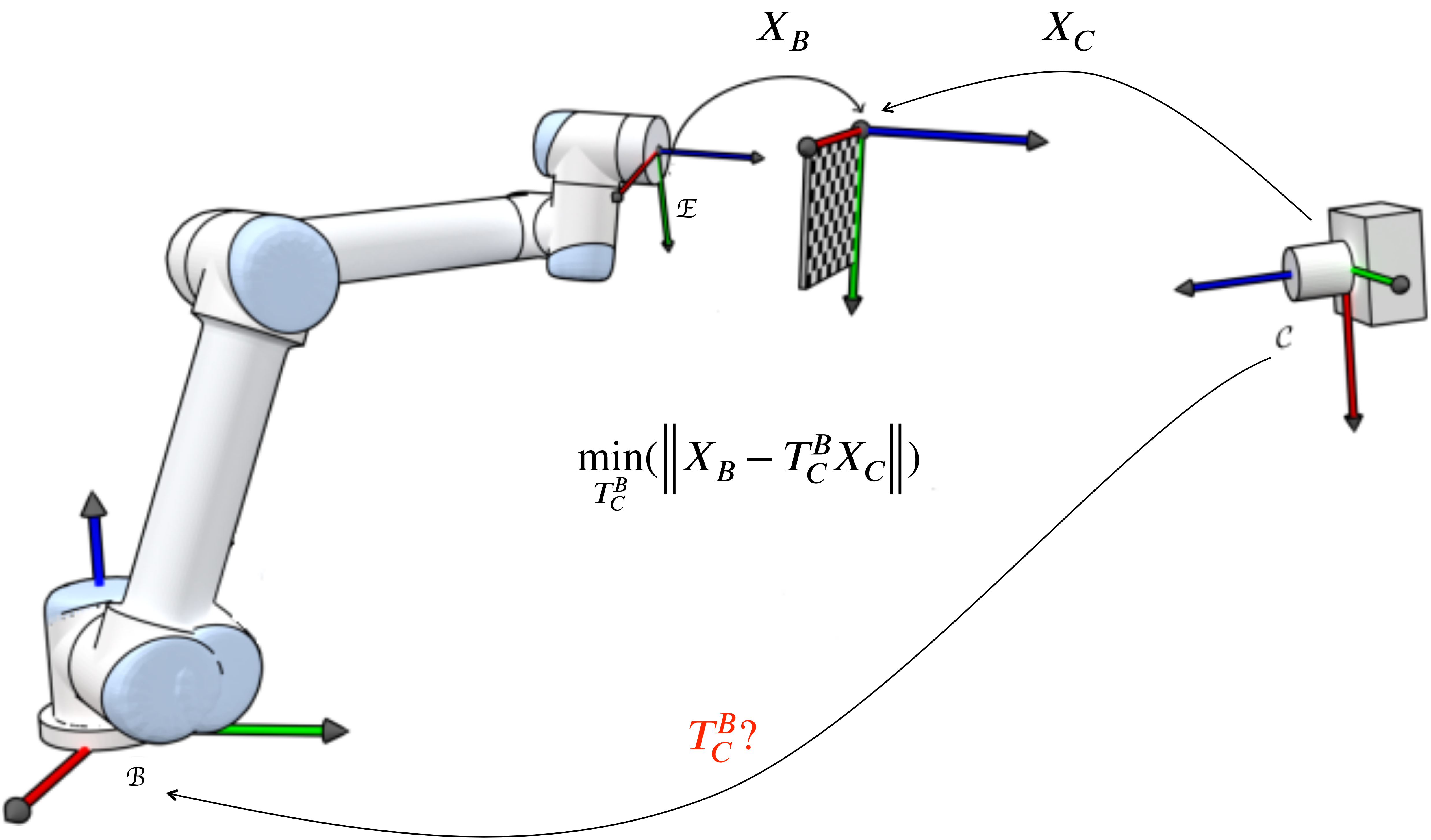
Slide from Dhiraj Gandhi.

Camera Calibration



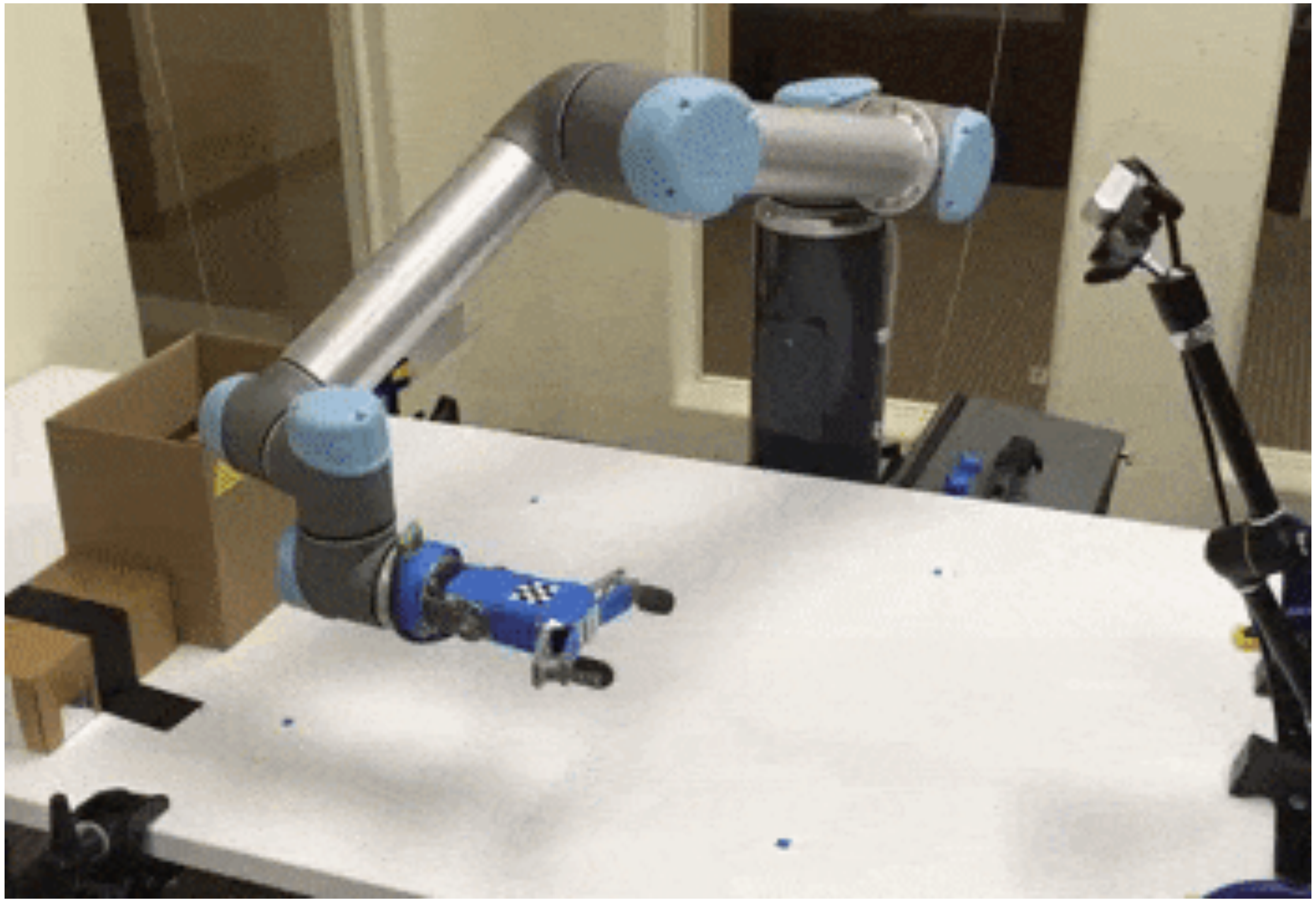
Slide from Dhiraj Gandhi.

Camera Calibration



Slide from Dhiraj Gandhi.

Camera Calibration

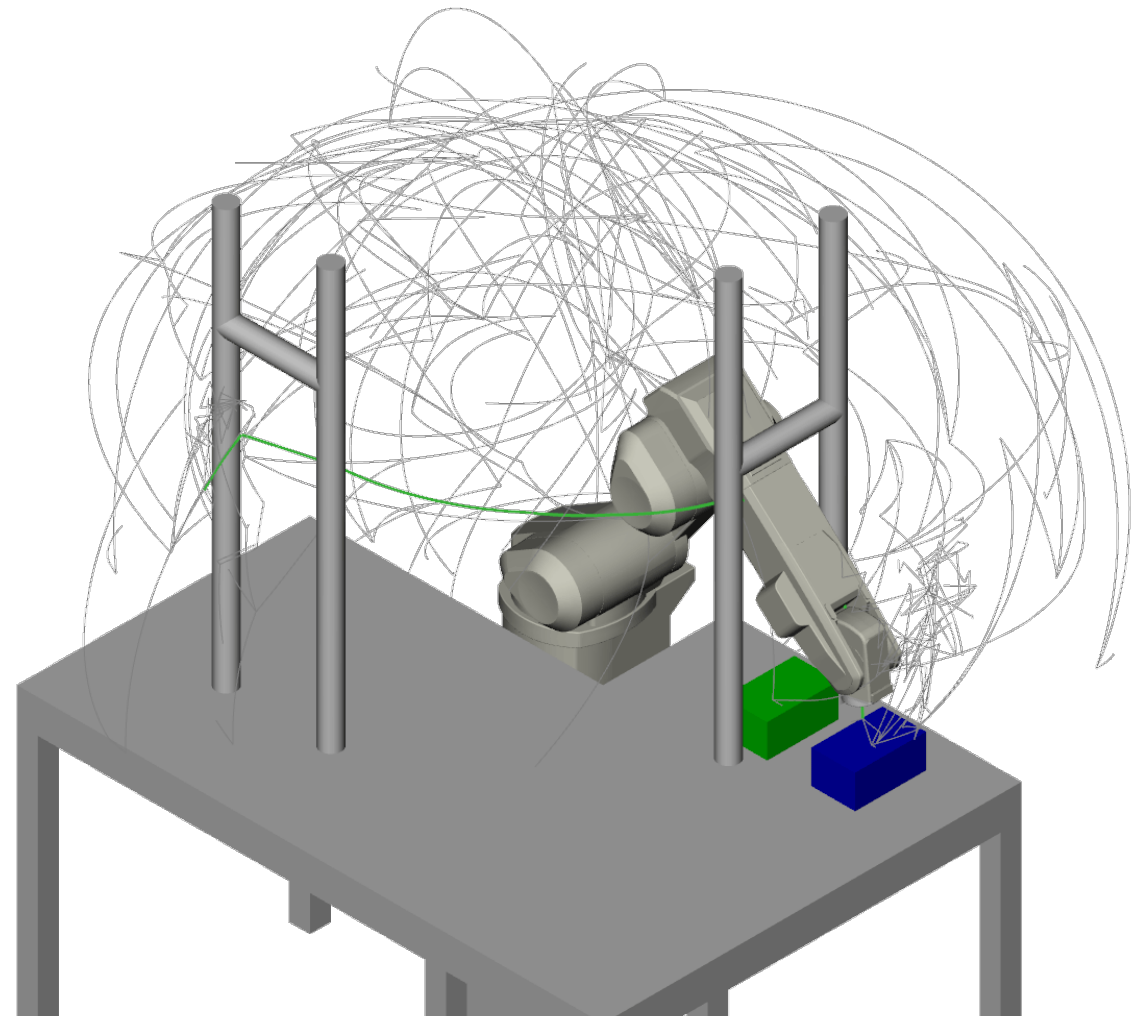


$$\min_{T_C^B} \sum_{i=1}^{n} \left\| X_B^i - T_C^B X_C^i \right\|$$

Slide from Dhiraj Gandhi.

Good Softwares

MoveIt!



```
<robot name="baxter">
  <link name="base">
  </link>
  <link name="torso">
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://baxter_description/meshes/torso/base_link.DAE'
      </geometry>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://baxter_description/meshes/torso/base_link_coll
      </geometry>
    </collision>
```

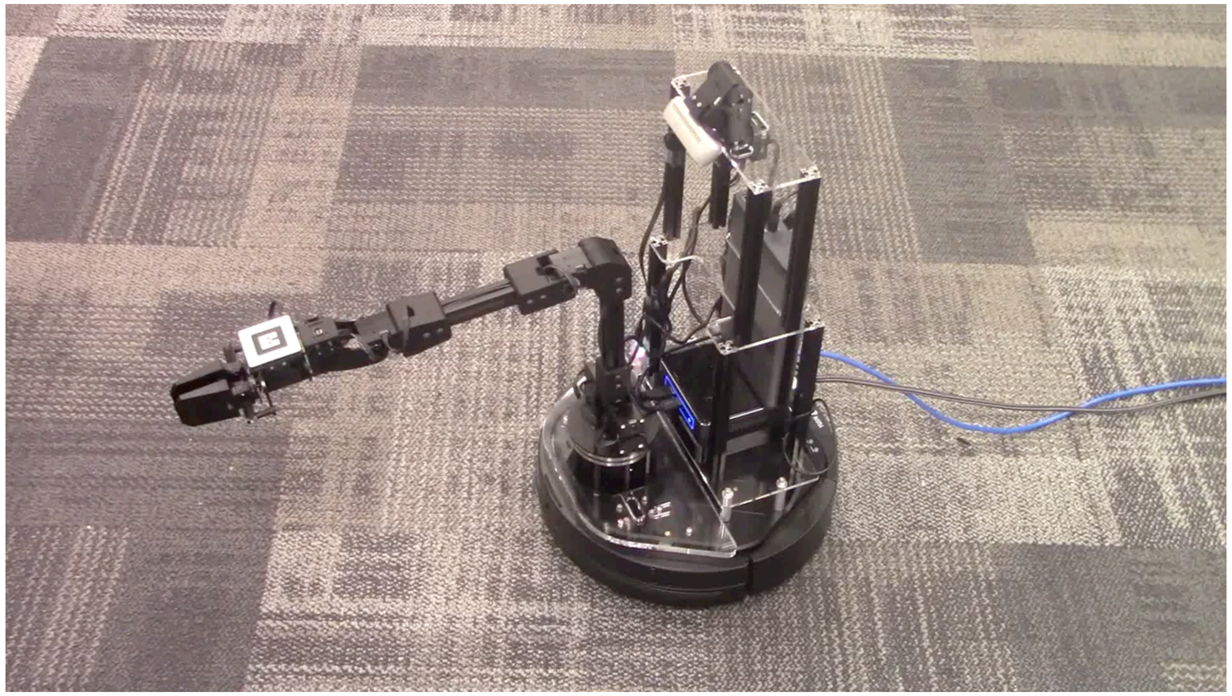
Slide from Dhiraj Gandhi.

MoveIt! Example

```python
target_poses = [
    {'position': np.array([0.28, 0.17, 0.22]),
     'pitch': 0.5,
     'numerical': False},
    {'position': np.array([0.28, -0.17, 0.22]),
     'pitch': 0.5,
     'roll': 0.5,
     'numerical': False}
]

robot.arm.go_home()

for pose in target_poses:
    robot.arm.set_ee_pose_pitch_roll(**pose)
    time.sleep(1)
robot.arm.go_home()
```



PyRobot: An Open-Source Robotics Framework for Research and Benchmarking.

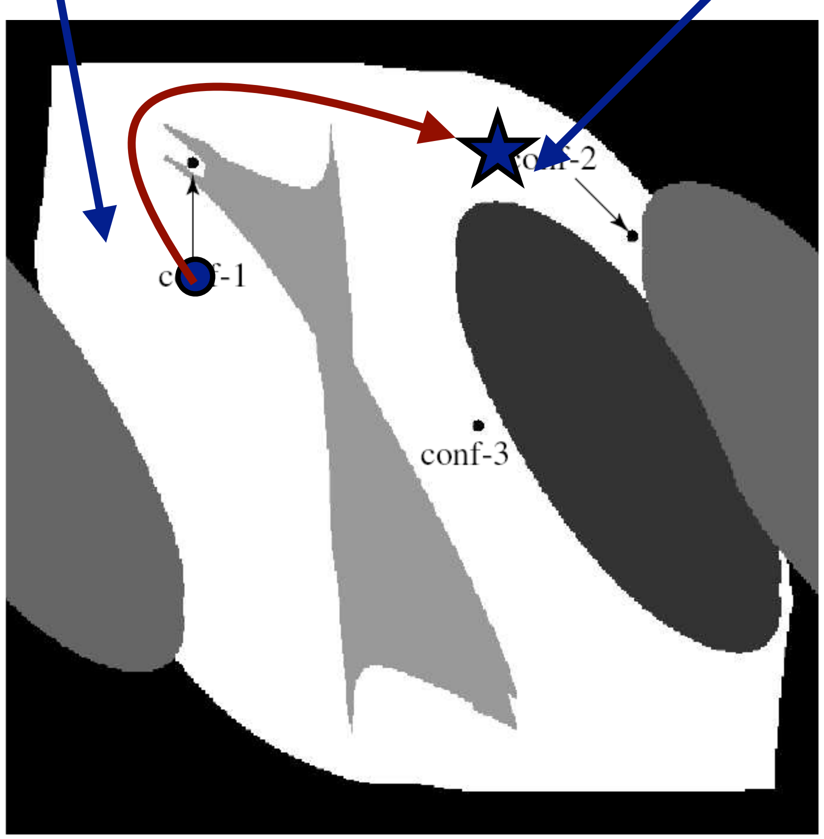
Robotics Review: How to move your robot?

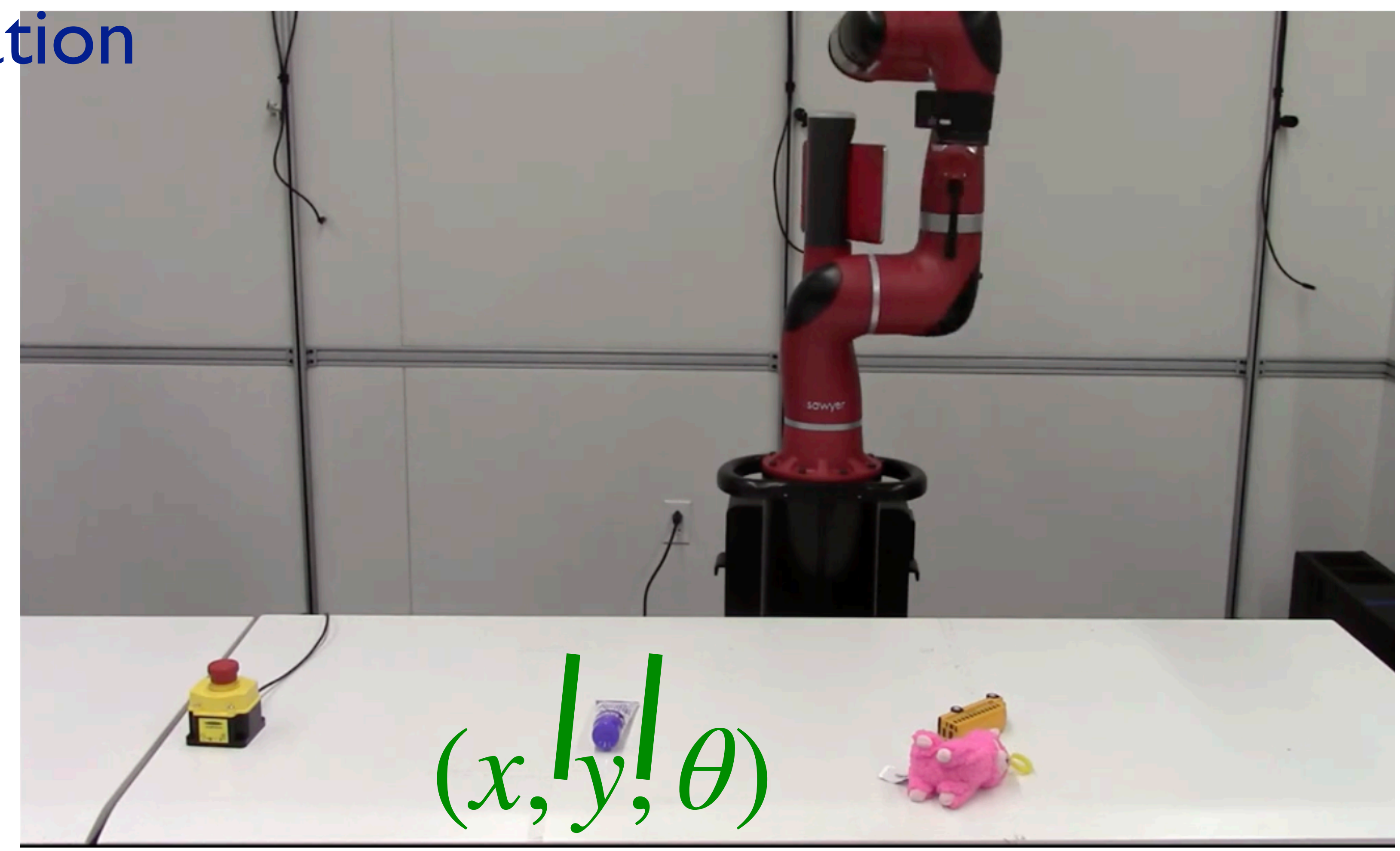
1. Task space to Configuration space
2. Configuration space trajectory
3. Trajectory execution

Configuration Space
Forward / Inverse Kinematics
Motion Planning
Optimal Control

Initial configuration

Desired configuration





Resources

Kris Hauser’s Robotic Systems Book
http://motion.cs.illinois.edu/RoboticSystems/

Pieter Abbeel's Advanced Robotics Course at Berkeley
https://people.eecs.berkeley.edu/~pabbeel/cs287-fa19/

Howie Choset's Robotic Motion Planning Course at CMU
https://www.cs.cmu.edu/~motionplanning/