Structure and Synthesis of Robot Motion

Introduction: Kinematics & Sampling-based Motion Planning

Subramanian Ramamoorthy
School of Informatics

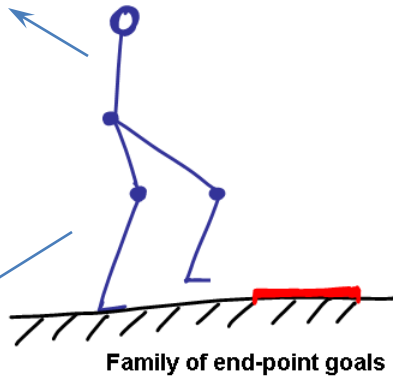
19 January, 2012

Describing Robot Motion

- Different aspects to the description
  - How does the robot move? How do we want to represent that?
  - Where does the robot move? How to represent environment?

- Consider a walking robot:

Can capture poses in a configuration space (a.k.a. joint space)

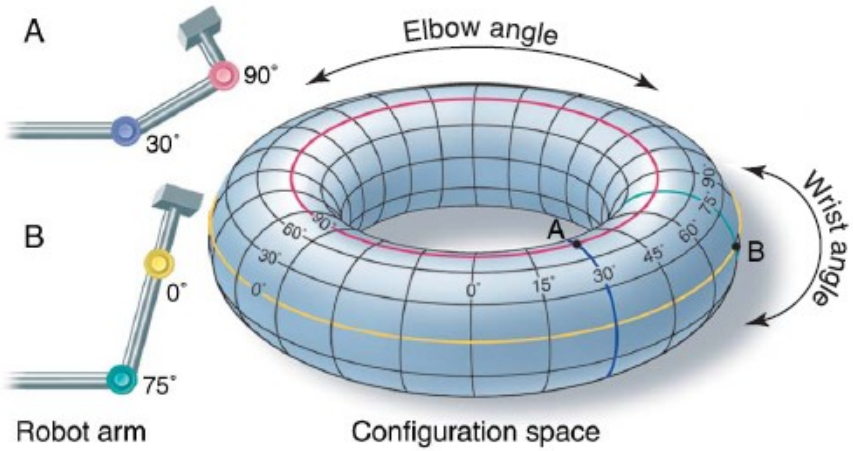


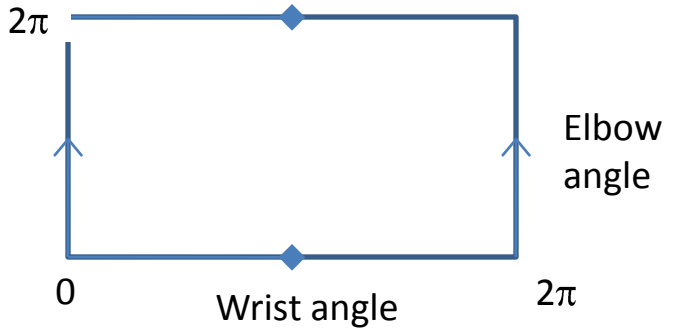
Some dynamics information is better understood in phase space

Environment/constraints would shape the geometry in the work/task space

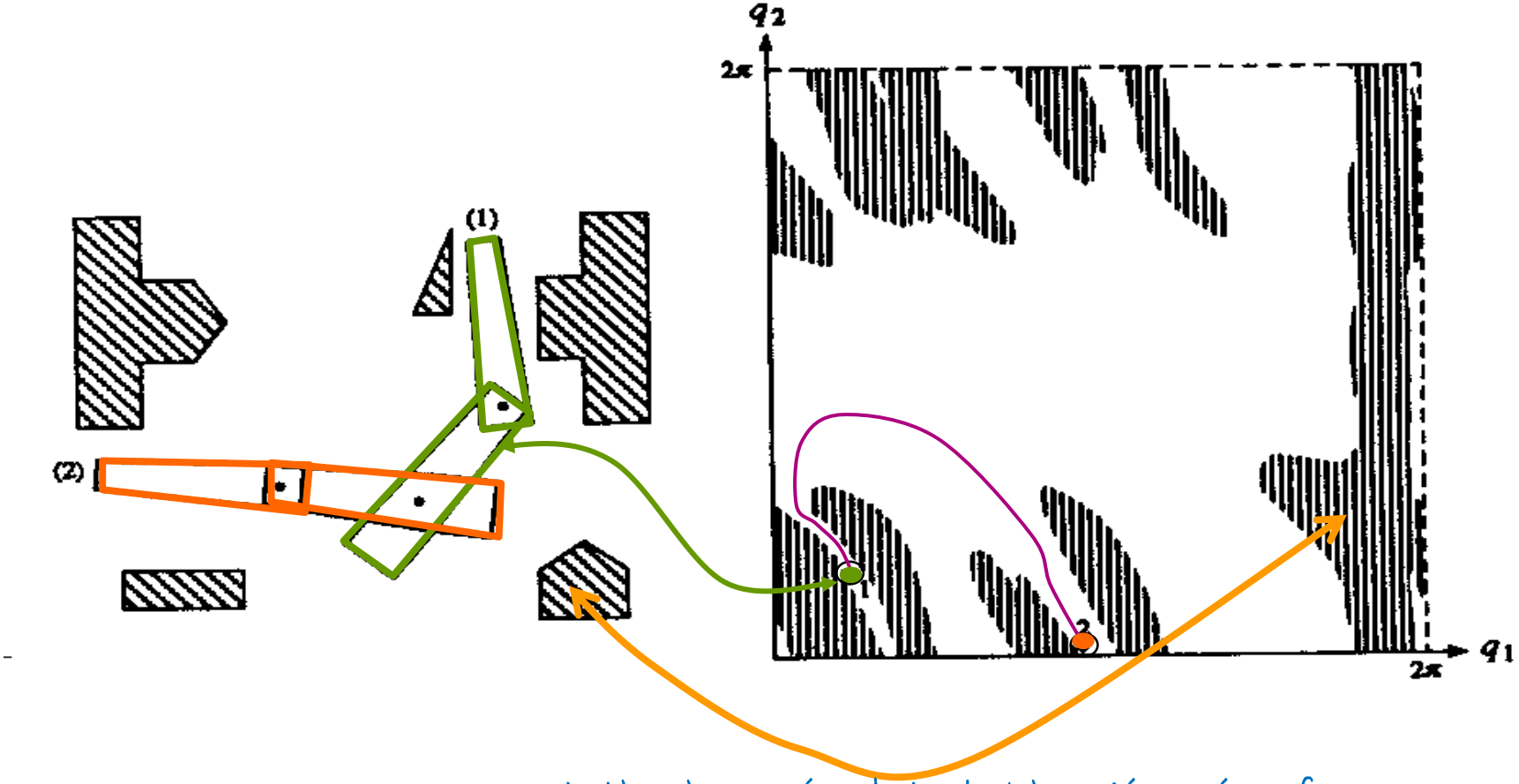
The Configuration Space

- A space describing the various configuration variables
- In the example, there are two rotational joints (each variable evolves along a circle, S^1)
  - Combined effect is that motion can be described as points on the torus



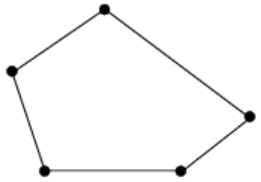


Relating C-Space and Workspace

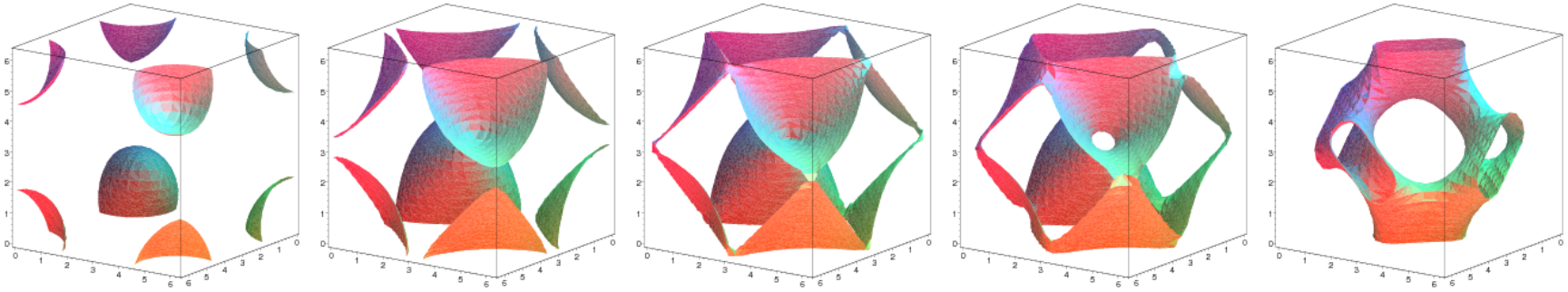


Note the change in obstacle 'shape' in going from one space to another. How do we efficiently compute this?

Configuration Spaces can be Complex



5-gonal linkage



(a) (b) (c) (d) (e)

5-gonal linkage $\mathcal{P}(1,1,1,1,l_5)$ with $l_5 \in \{3, 2.1, 2, 1.9, 1\}$

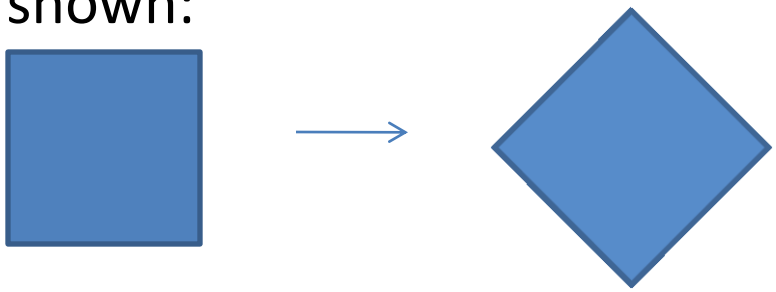
What is Kinematics?

- Robot kinematics: In a kinematic analysis the position, velocity and acceleration of all links are calculated without considering forces that cause this motion.
  - As opposed to dynamics - which studies relation between motion of objects and its causes.

- Robot kinematics deals with aspects of redundancy, collision avoidance and singularity avoidance.

Start from Basics: Simplest Example

- Consider a 2-dim object on a plane, goes from one pose to another as shown:



- To describe it (in a computer program), keep track of
  - Translation of a canonical point, say, the center – (x,y)
  - Angle by which object is rotated (with appropriate convention for axis and direction) - θ
- If I have many objects, I'll need to pick a global origin and describe them all in terms of corresponding (x, y, θ)

2-*dim* Motion

Translation:

Every point is mapped as $(x, y) \mapsto (x + x_t, y + y_t)$

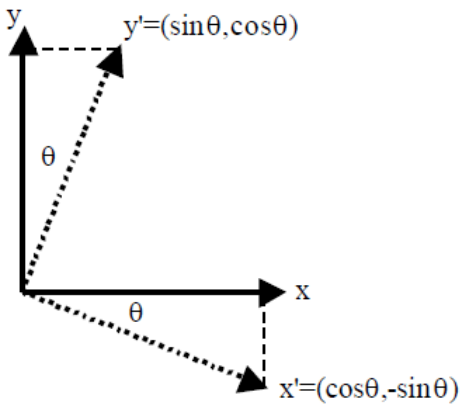
Rotation:

Every point is mapped as $(x, y) \mapsto (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$

This can we written in terms of a *rotation matrix*,

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

So, every point is transformed as $\begin{pmatrix} x \\ y \end{pmatrix} \mapsto R(\theta) \begin{pmatrix} x \\ y \end{pmatrix}$



Such equations can describe two things:

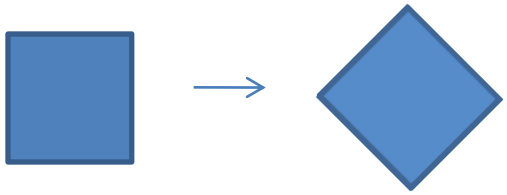
- Change of coordinate frames between points in robot
- New position of an object if it performs a motion in the environment

2-*dim* Motion

Translation and Rotation:

This involves a 3×3 *homogeneous transformation matrix*, where rotation is *followed* by a translation:

$$T(x_t, y_t, \theta) = \begin{pmatrix} \cos\theta & -\sin\theta & x_t \\ \sin\theta & \cos\theta & y_t \\ 0 & 0 & 1 \end{pmatrix}$$



Describing 3-*dim* motion

Translation:

Every point is mapped as $(x, y, z) \mapsto (x + x_t, y + y_t, z + z_t)$

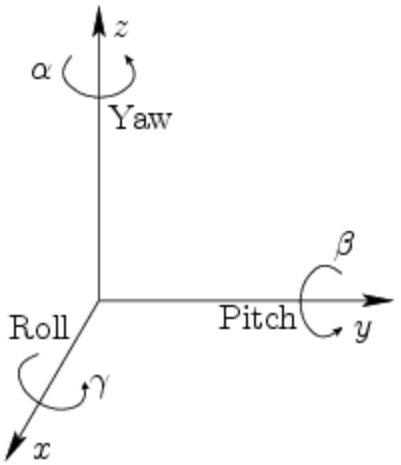
Rotation:

Different rotation matrix for each axis.

Yaw: $R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$

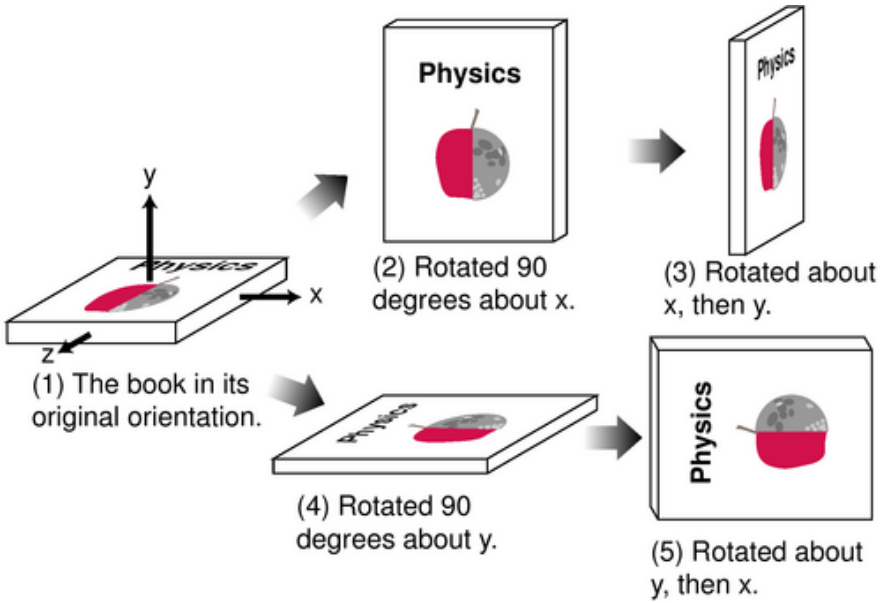
Pitch: $R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}$

Roll: $R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix}$



(Non)-commutativity

In 3-*dim* (and beyond), interesting things begin to happen:



You need to know the order of operations to be able to fully describe the result!

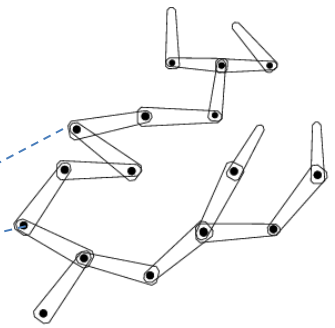
Combined 3-dim Rotations

$$T = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & x_t \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & y_t \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & z_t \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Which corresponds to: Roll by γ; pitch by β; yaw by α; translate by (x_t, y_t, z_t).

If a robot/object consists of many links/parts, we could compute position of any end with respect to a base via an appropriate sequence of transformations:

$$T = T_0 T_1 \dots T_m$$

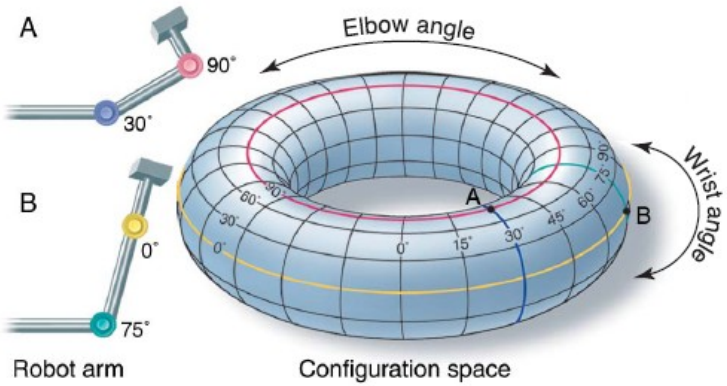


Where are these two points w.r.t. each other?

A Question

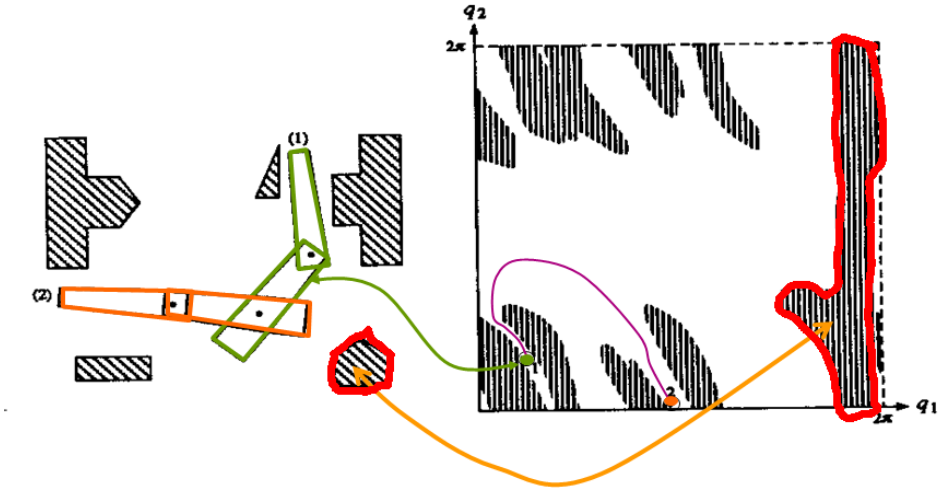
- Suppose you want to hold some things fixed in your robot while the rest of the mechanism does its thing…

  e.g., you want to hold a glass of water (hence your ‘end effector’ position) while moving the hand/glass to your mouth
- This happens in many settings:
  - Constraint (wiping windows)
  - Dynamics (Do not fall/slip)
  - Task spec. (‘standing’ poses)

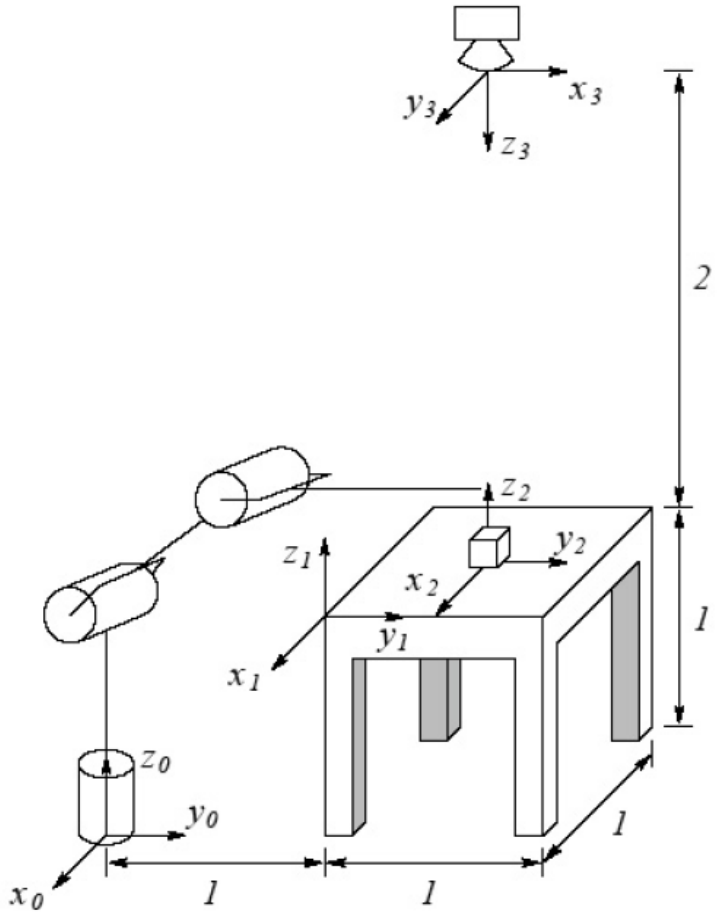


Describing Subspaces

- The kind of restrictions in the previous slide correspond to subspaces
- A linear subspace would be a hyperplane
- In general, when we translate between spaces, the restrictions take on a more general form



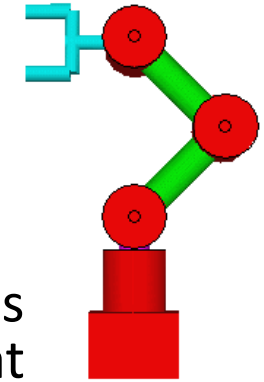
A Typical Scenario



- Many robotics problems involve a workspace goal that needs to be achieved through joint space actions
- Translating between the two coordinate systems requires thinking through a sequence of coord. Frames
- In the figure:
  - How does object motion translate to robot manipulator motion?

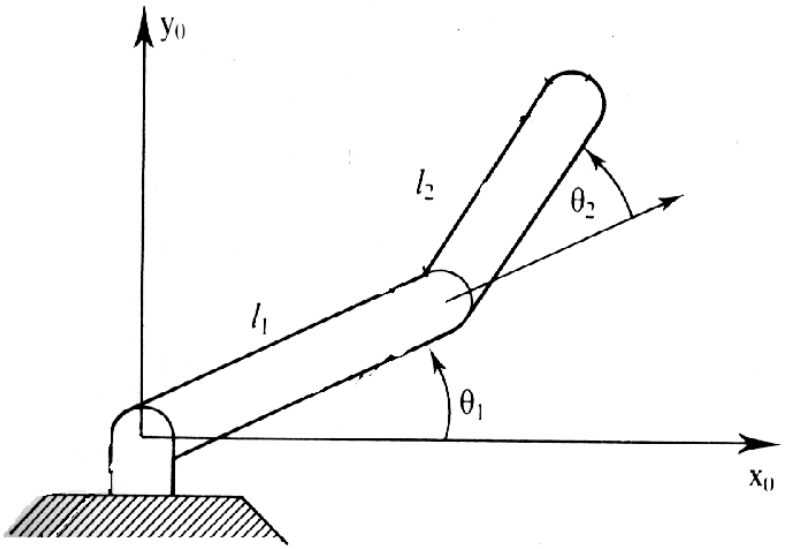
Two Aspects of Kinematics

- Forward Kinematics
  - Given joint angles, compute the transformation between world & gripper coordinates
  - Relatively straightforward



- Inverse Kinematics
  - Given the transformation between world coordinates and an arbitrary frame, compute the joint angles that would line your gripper coordinates up with that frame.
  - More involved than forward case, as will see

Forward Kinematics



Scenario:

You have a robotic arm that starts out aligned with the x_o-axis.

You tell the first link to move by θ_1 and the second link to move by θ_2

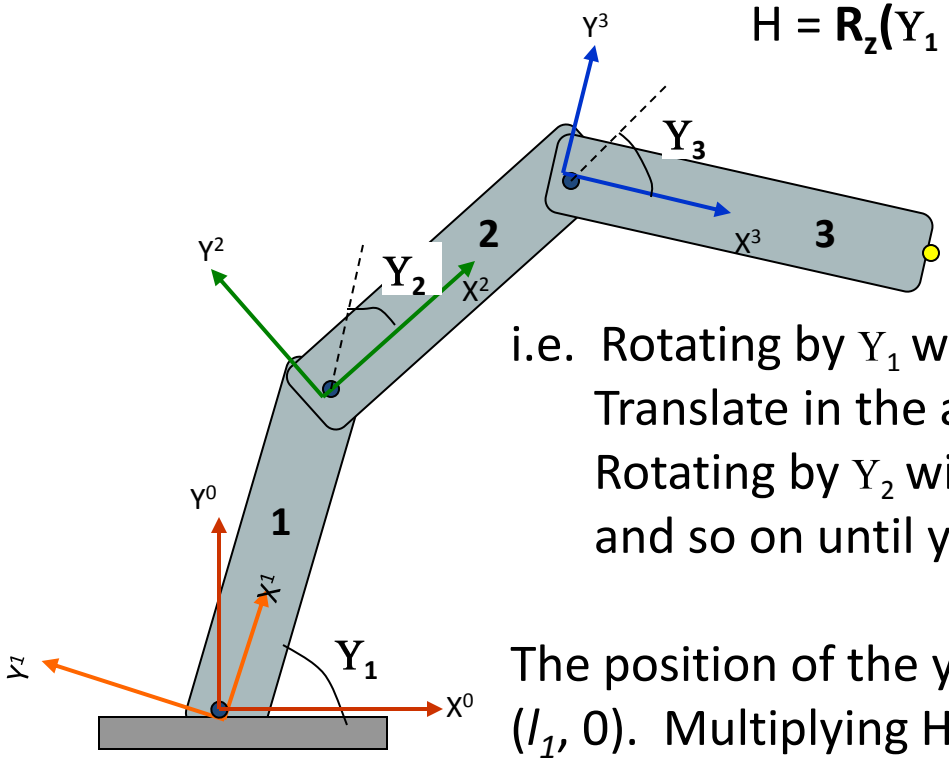
Question:

What is the position of the end of the robotic arm?

Solution:

Use transformation matrices from earlier slides

Locate the Yellow Dot in Base Frame



Translation, along x^2, by l_2

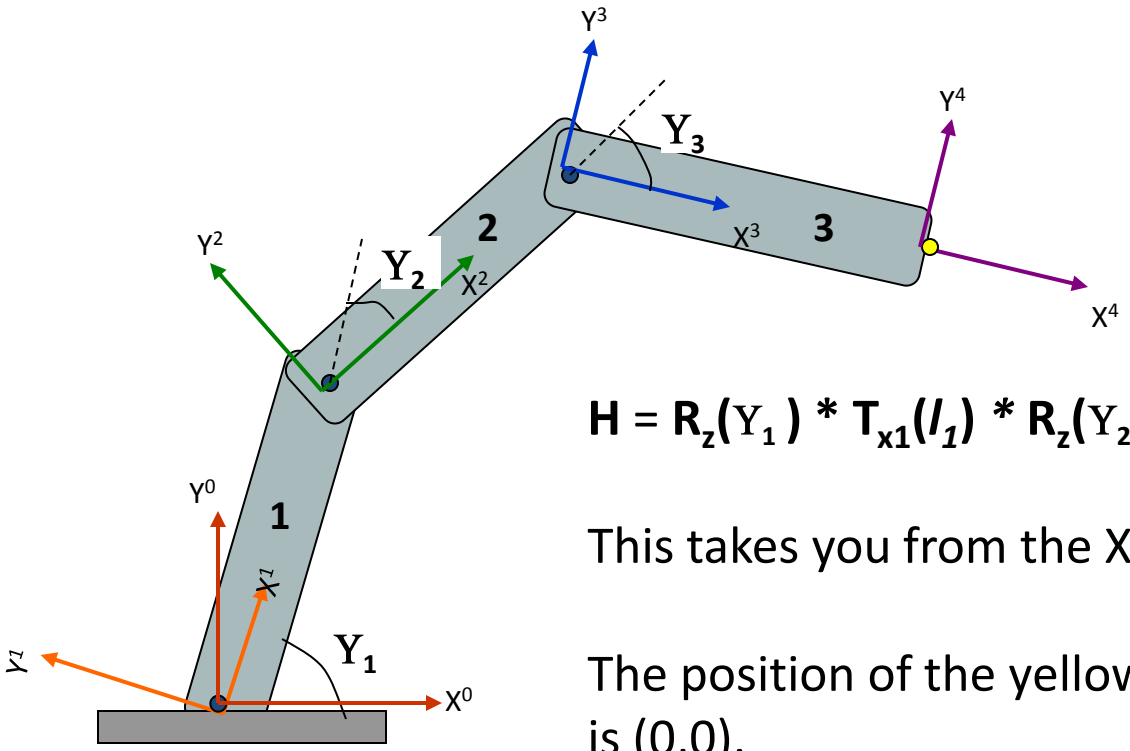
$$H = R_z(\Upsilon_1) * T_{x1}(l_1) * R_z(\Upsilon_2) * T_{x2}(l_2) * R_z(\Upsilon_3)$$

Rotation, around z, by Y_2

i.e. Rotating by Υ_1 will put you in the X^1Y^1 frame.
Translate in the along the X^1 axis by l_1.
Rotating by Υ_2 will put you in the X^2Y^2 frame.
and so on until you are in the X^3Y^3 frame.

The position of the yellow dot relative to the X^3Y^3 frame is (l_1, 0). Multiplying H by that position vector will give you the coordinates of the yellow point relative the the X^0Y^0 frame.

Variation: New Coordinate Frame at the Dot



$$H = R_z(\Upsilon_1) * T_{x1}(l_1) * R_z(\Upsilon_2) * T_{x2}(l_2) * R_z(\Upsilon_3) * T_{x3}(l_3)$$

This takes you from the X^0Y^0 frame to the X^4Y^4 frame.

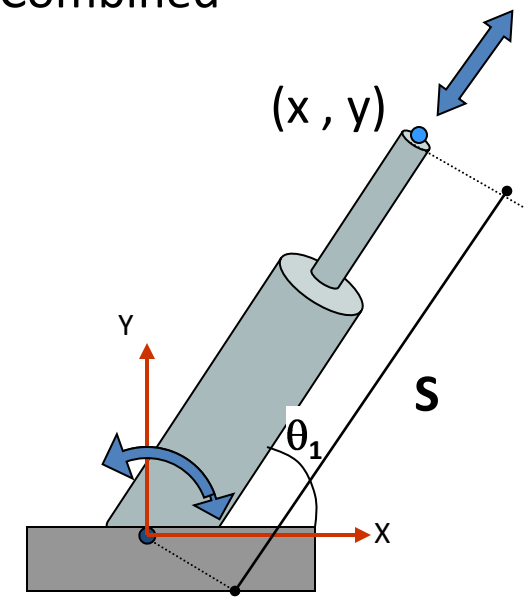
The position of the yellow dot relative to the X^4Y^4 frame is (0,0).

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = H \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Notice that multiplying by the (0,0,0,1) vector will equal the last column of the H matrix.

Inverse Kinematics: Simplest Example

Revolute and
Prismatic Joints
Combined



Find Angle:

$$\theta = \arctan\left(\frac{y}{x}\right)$$

More Specifically:

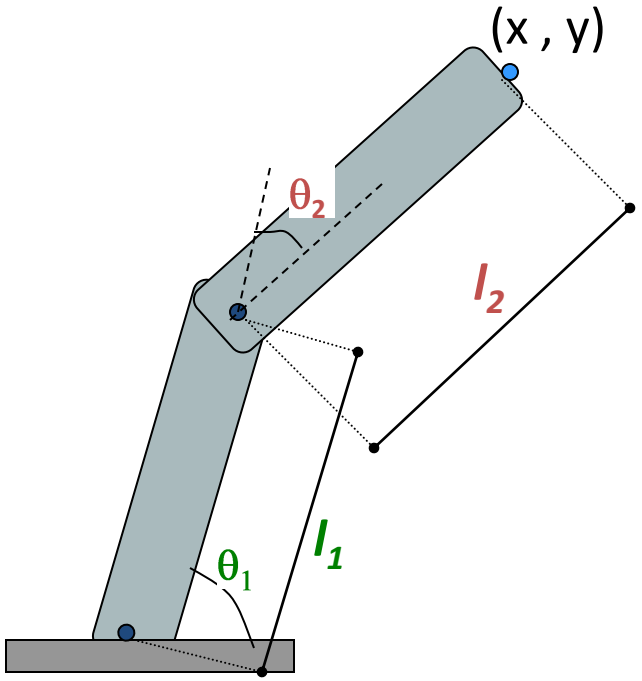
$$\theta = \arctan 2\left(\frac{y}{x}\right)$$

arctan2() specifies that it's in the first quadrant

Finding **S**:

$$S = \sqrt{(x^2 + y^2)}$$

Bit More Complex: 2-link Manipulator



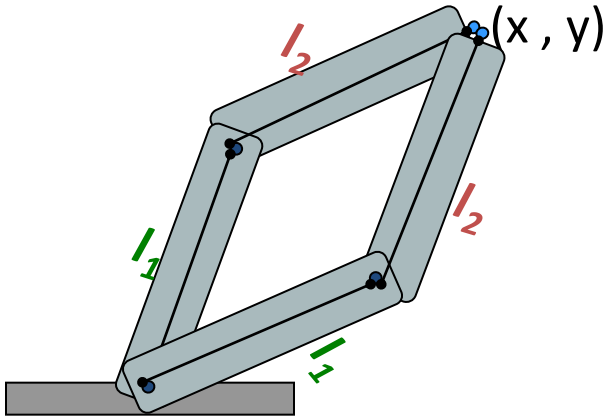
Given: l_1, l_2, x, y

Find: θ_1, θ_2

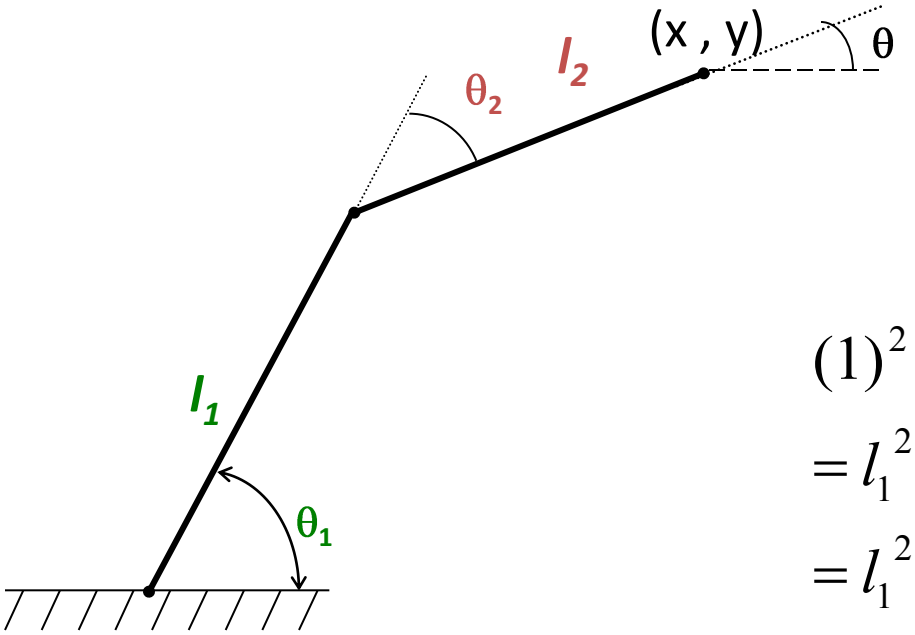
Redundancy:

A unique solution to this problem does not exist - two solutions are possible.

Sometimes no solution is possible.



Solving for the Joint Angles



$$c_1 = \cos\theta_1$$

$$c_{1+} = \cos(\theta_2 + \theta_1)$$

$$(1)\ x = l_1 c_1 + l_2 c_{1+}$$

$$(2)\ y = l_1 s_1 + l_2 \sin_{1+}$$

$$(3)\ \theta = \theta_1 + \theta_2$$

$$(1)^2 + (2)^2 = x^2 + y^2 =$$

$$= l_1^2 + l_2^2 + 2l_1 l_2 \left[c_1 (c_{1+2}) + s_1 (\sin_{1+2}) \right]$$

$$= l_1^2 + l_2^2 + 2l_1 l_2 c_2$$

$$\therefore \theta_2 = \arccos\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2} \right)$$

Solving for Joint Angles, Contd.

$$
\begin{aligned}
x &= {}_1 c_1 + {}_2 c_{1+} \\
&= {}_1 c_1 + {}_2 c_1 c_2 - {}_2 s_1 s_2 \\
&= {}_1(l_1 + {}_2 c_2) - {}_1(l_2 s_2) \\
y &= {}_1 s_1 + {}_2 s_{1+} \\
&= {}_1 s_1 + {}_2 s_1 c_2 + {}_2 s_2 c_1 \\
&= {}_1(l_2 s_2) + {}_1(l_1 + {}_2 c_2)
\end{aligned}
$$

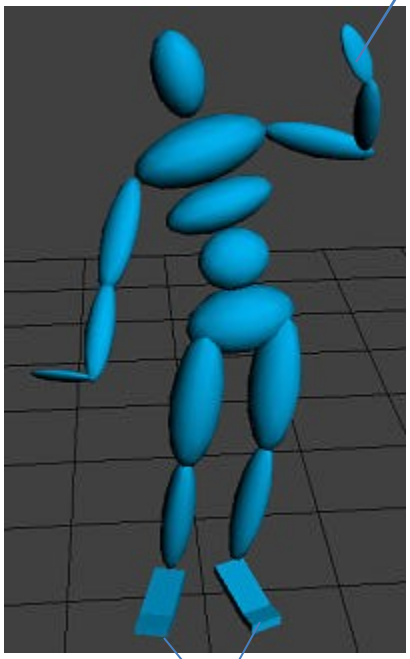
$$
y = \frac{x + s_1(l_2 s_2)}{(l_1 + l_2 c_2)}(l_2 s_2) + {}_1(l_1 + l_2 c_2)
$$

$$
= \frac{1}{(l_1 + l_2 c_2)}\left(x l_2 s_2 + {}_1(l_1^2 + l_2^2 + l_1 l_2 c_2) \right)
$$

$$
s_1 = \frac{y(l_1 + l_2 c_2) - x l_2 s_2}{x^2 + y^2} \qquad \theta_1 = \arcsin\left(\frac{y(l_1 + l_2 c_2) - x l_2 s_2}{x^2 + y^2} \right)
$$

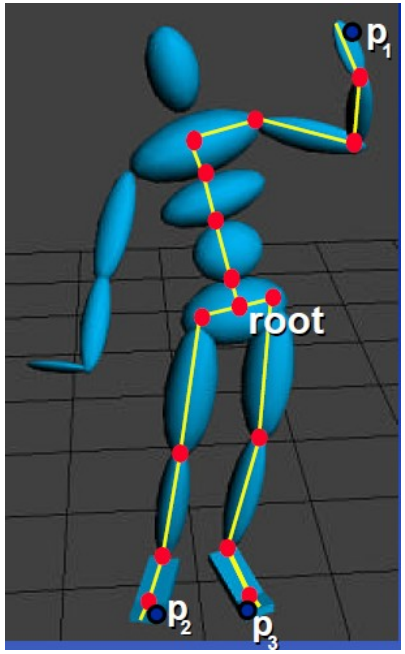
Things Can Get Complicated…

Hands should follow the mouse



Keep feet fixed

This is still within our model:



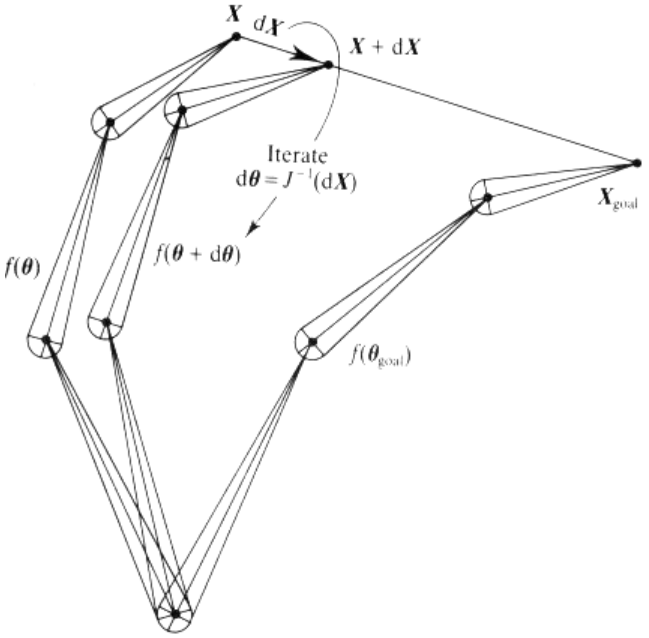
But, we'd like to avoid tedious calculations like in last slide

Use of the Jacobian – Computing *Velocities*

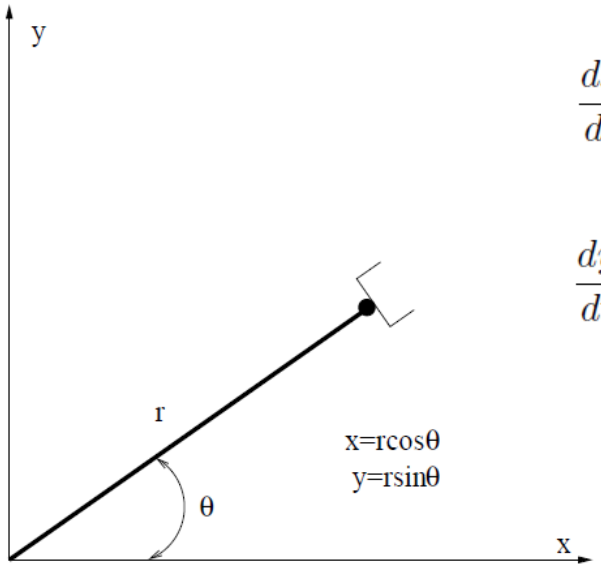
- Chain of links maps a set of input variables (Joint Angles) to end effecter variables: $X = F(\theta$
- The Jacobian is defined as:

$$J(\theta) = \frac{\partial F(\theta)}{\partial \theta}$$

- A transformation from joint velocities to end-point velocities: $V = \dot{X} = J(\theta)\dot{\theta}$



Jacobian: A Simple Example



$$\frac{dx}{dt} = \frac{\partial(r\,cos\theta)}{\partial r}\frac{dr}{dt} + \frac{\partial(r\,cos\theta)}{\partial \theta}\frac{d\theta}{dt} \quad ==> \quad \dot{x} = cos\theta\dot{r} - r sin\theta\dot{\theta}$$

$$\frac{dy}{dt} = \frac{\partial(r\,sin\theta)}{\partial r}\frac{dr}{dt} + \frac{\partial(r\,sin\theta)}{\partial \theta}\frac{d\theta}{dt} \quad ==> \quad \dot{y} = sin\theta\dot{r} + r\,cos\theta\dot{\theta}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J \begin{bmatrix} \dot{\theta} \\ \dot{r} \end{bmatrix} \quad where \quad J = \begin{bmatrix} -r\,sin\,\theta & cos\,\theta \\ r\,cos\,\theta & sin\,\theta \end{bmatrix}$$

Using the Jacobian Inverse

- Use task level goals to figure out how end effecter should move - *V*
- Solve for the joint angle velocities:

$$V = \dot{r} = J(\theta)\dot{\theta}$$

- Some pitfalls:
  - J might be singular
  - Overdetermined system

Concept of **Pseudoinverse**:

$$V = J\dot{\theta}$$

$$J'V = J'J\dot{\theta}$$

$$(J'J)^{-} J'V = \dot{\theta}$$

$$J^{+}V = \dot{\theta}$$

$$\text{where } J^{+} = (J'J)^{-} J' = J'(JJ')^{-}$$

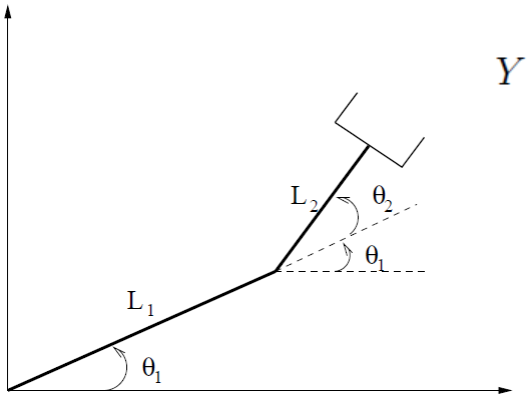
$$\left[J^{-1}\right]\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}\dot{\theta}\\ \dot{r}\end{bmatrix} \quad where \quad J^{-1} = \begin{bmatrix}\frac{-sin\,\theta}{r} & \frac{cos\,\theta}{r}\\ cos\,\theta & sin\,\theta\end{bmatrix}$$

Jacobian of the 2-link Arm

$$T_1^0 = \begin{bmatrix} C_1 & -S_1 & 0 & C_1L_1 \\ S_1 & C_1 & 0 & S_1L_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_2^1 = \begin{bmatrix} C_2 & -S_2 & 0 & C_2L_2 \\ S_2 & C_2 & 0 & S_2L_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_2^0 = \begin{bmatrix} C_{12} & -S_{12} & 0 & C_1L_1 + L_2C_{12} \\ S_{12} & C_{12} & 0 & S_1L_1 + L_2S_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$X = C_1L_1 + L_2C_{12}\,;\ \frac{\partial X}{\partial t} = \frac{\partial(C_1L_1 + L_2C_{12})}{\partial\Theta_1}\frac{\partial\Theta_1}{\partial t} + \frac{\partial(C_1L_l + L_2C_{12})}{\partial\Theta_2}\frac{\partial\Theta_2}{\partial t}$$

$$\dot{X} = (-S_1L_1 - L_2S_{12})\dot{\Theta}_1 - L_2S_{12}\dot{\Theta}_2$$



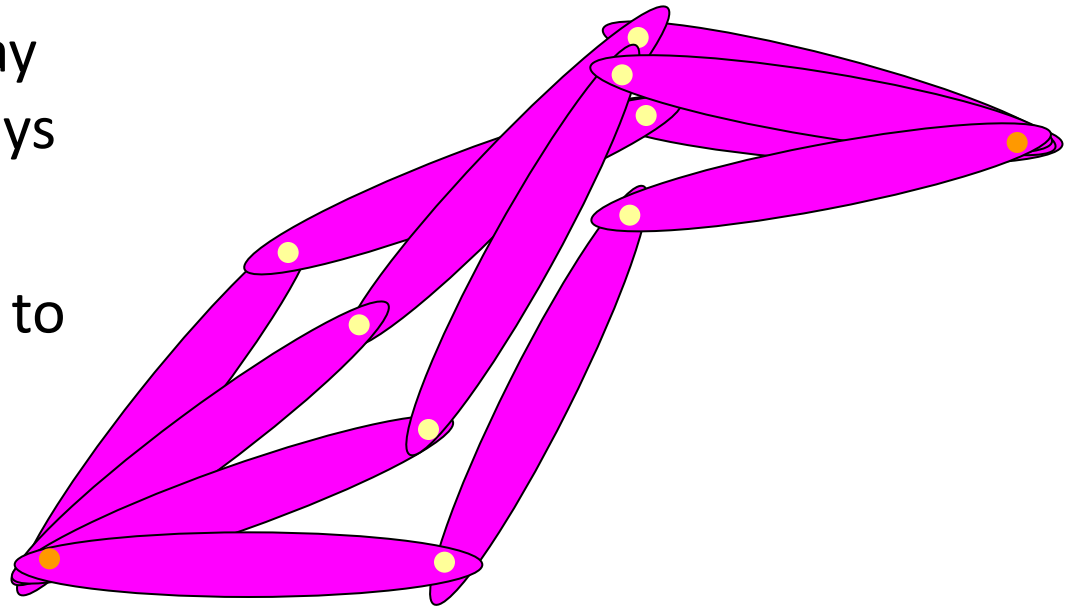
$$Y = S_1L_1 + L_2S_{12}\,;\ \frac{\partial Y}{\partial t} = \frac{\partial(S_1L_1 + L_2S_{12})}{\partial\Theta_1}\frac{\partial\Theta_1}{\partial t} + \frac{\partial(S_1L_1 + L_2S_{12})}{\partial\Theta_2}\frac{\partial\Theta_2}{\partial t}$$

$$\dot{Y} = (C_1L_1 + L_2C_{12})\dot{\Theta}_1 + L_2C_{12}\dot{\Theta}_2$$

$$\textit{and} \quad \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} -S_1L_1 - L_2S_{12} & -L_2S_{12} \\ C_1L_1 + L_2C_{12} & L_2C_{12} \end{bmatrix} \begin{bmatrix} \dot{\Theta}_1 \\ \dot{\Theta}_2 \end{bmatrix}$$

Regularizing the Inverse

- While the pseudoinverse will yield *a* solution, it may not be the *right* one always
- What are the criteria?
- One way to handle this is to define additional optimisation criteria:



$$\dot{\theta} = J^{+}(\theta)V - (I - J^{+}J)N\frac{\partial T(\theta)}{\partial \theta}$$

N : Orthonormal basis of null space

$T(\theta)$: "Goodness" function

(Kinematic) Motion Planning

Once we know how to compute forward/inverse kinematics, we are in a position to ask questions about planning

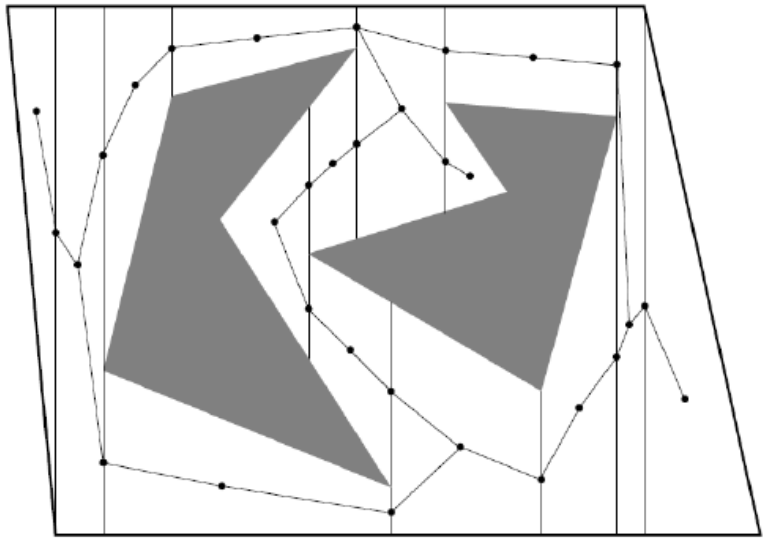
1. Given a start and end state, how to move between them?

2. How to represent external world and imposed constraints?
   - How much detail do you need?

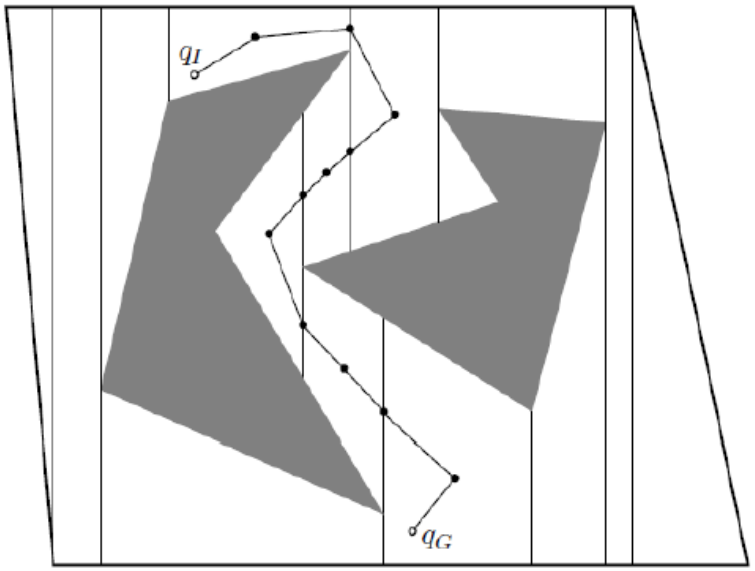
Roadmaps

Collapse description down to a (topological) relation between landmarks that suffices for planning purposes

A Combinatorial Roadmap



The roadmap derived from the vertical cell decomposition.



An example solution path.

Computations can quickly get very expensive!

Probabilistic Roadmaps

- Exploit the following observation:

  It is *cheap* to check if a single robot configuration is in free space or not
- Note this assumption: *need collision detection ability*
- So, by coarse sampling, one may compute nodes of a roadmap that are in free space
- Fine sampling (local planner) picks edges connecting nodes

- At 'query' time, connect start and goal points to nearest node in roadmap and perform path planning as before

Generic Procedure for Sampling-based Planning

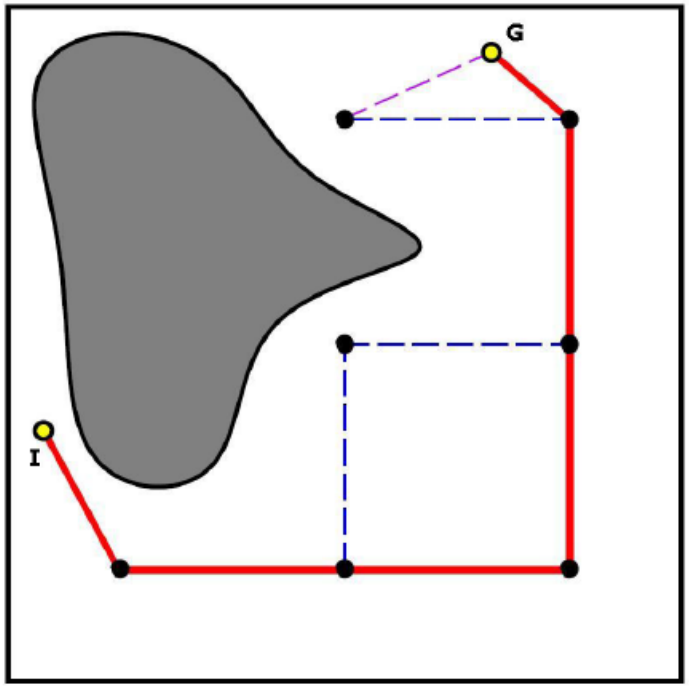
1. Initialize an undirected search graph (V, E), where V contains q_I, or q_G, or both, and E is empty.

2. Choose a vertex $q \in V$ for expansion. Construct a local path $\tau_e : [0, 1] \mapsto \mathcal{C}_{free}$ such that $\tau(0) = q$ and $\tau(1) = q_{new}$ for some $q_{new} \in \mathcal{C}_{free}$.

3. If $q_{new} \notin V$ then add q_{new} to V. Also add local path τ_e to E.

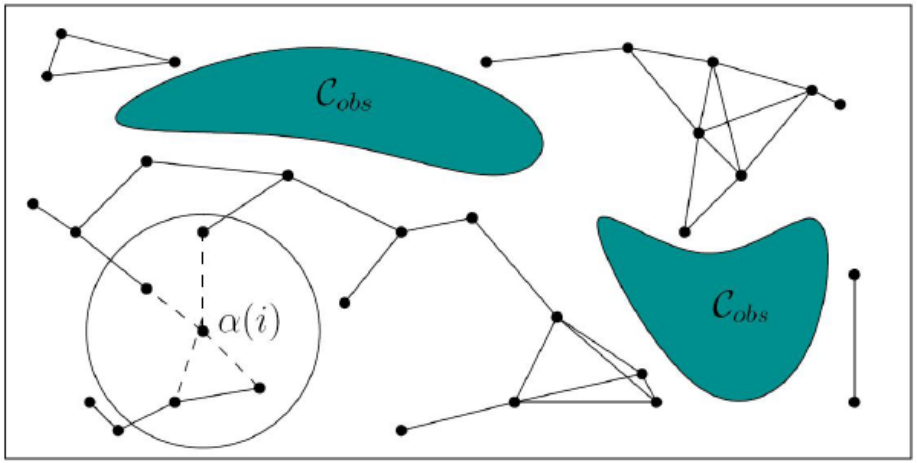
4. Search for a global path in (V, E) that connects q_I with q_G. If found (or some termination condition is satisfied) then terminate, else go to 2.

Generic Procedure, Visually



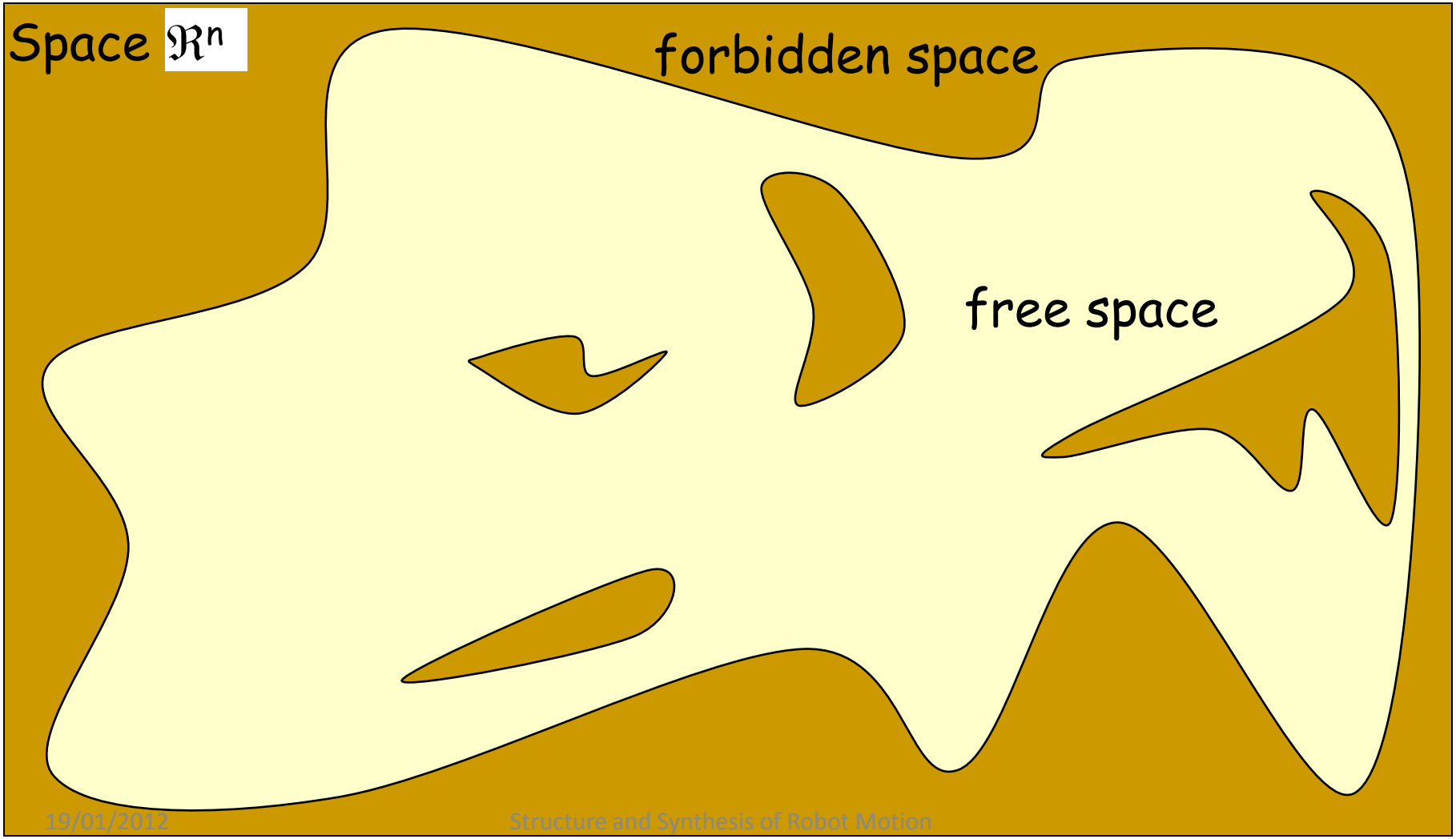
An illustration of motion planning as incremental graph search.

Sampling-based Roadmaps



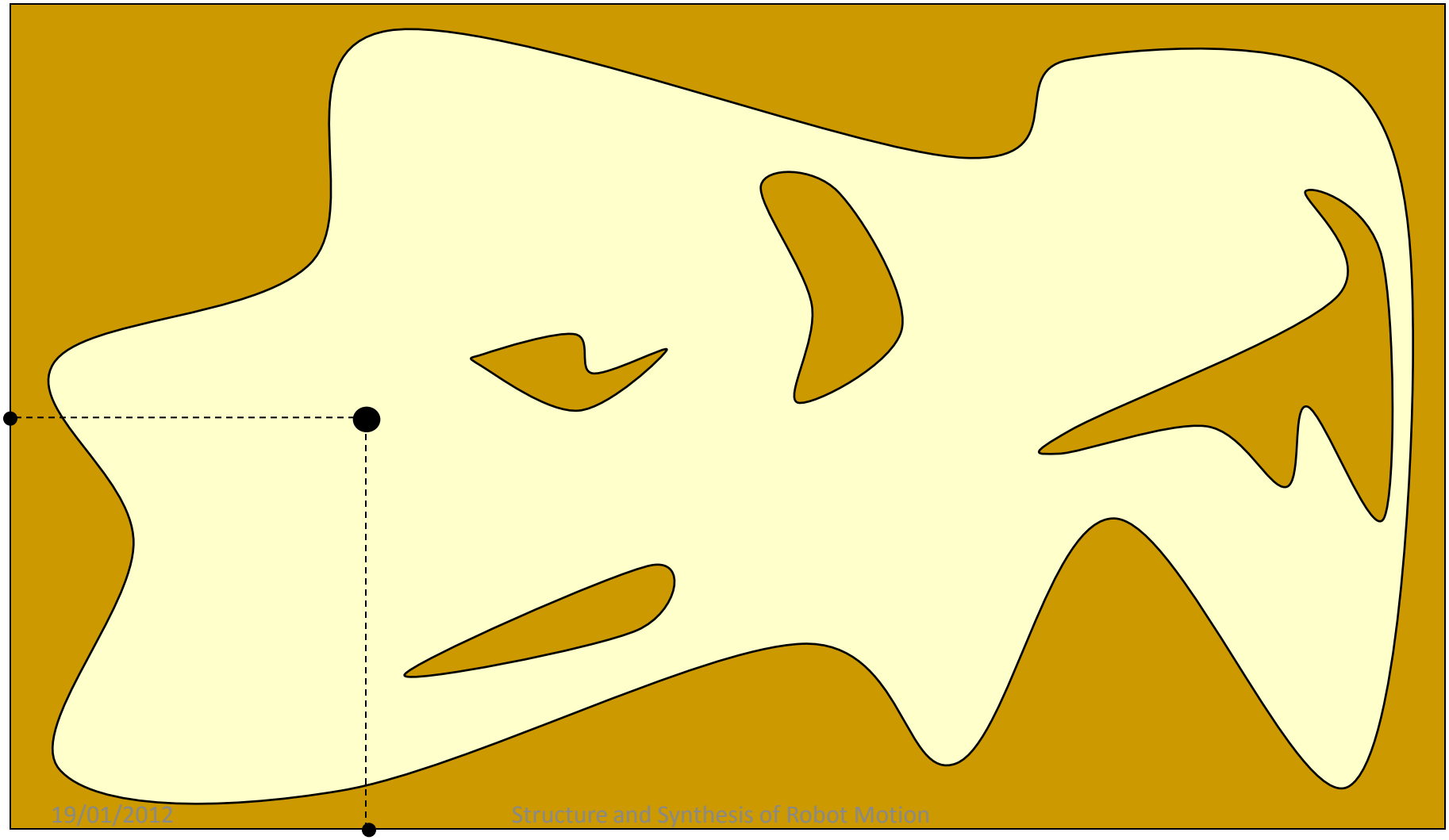
- A roadmap can be used for multiple queries. In a sampling-based roadmap, $\alpha(i)$ is connected to all its neighbors in the graph.
- In a probabilistic roadmap (PRM), $\alpha(i)$ is a random sequence.

Probabilistic Roadmap Construction



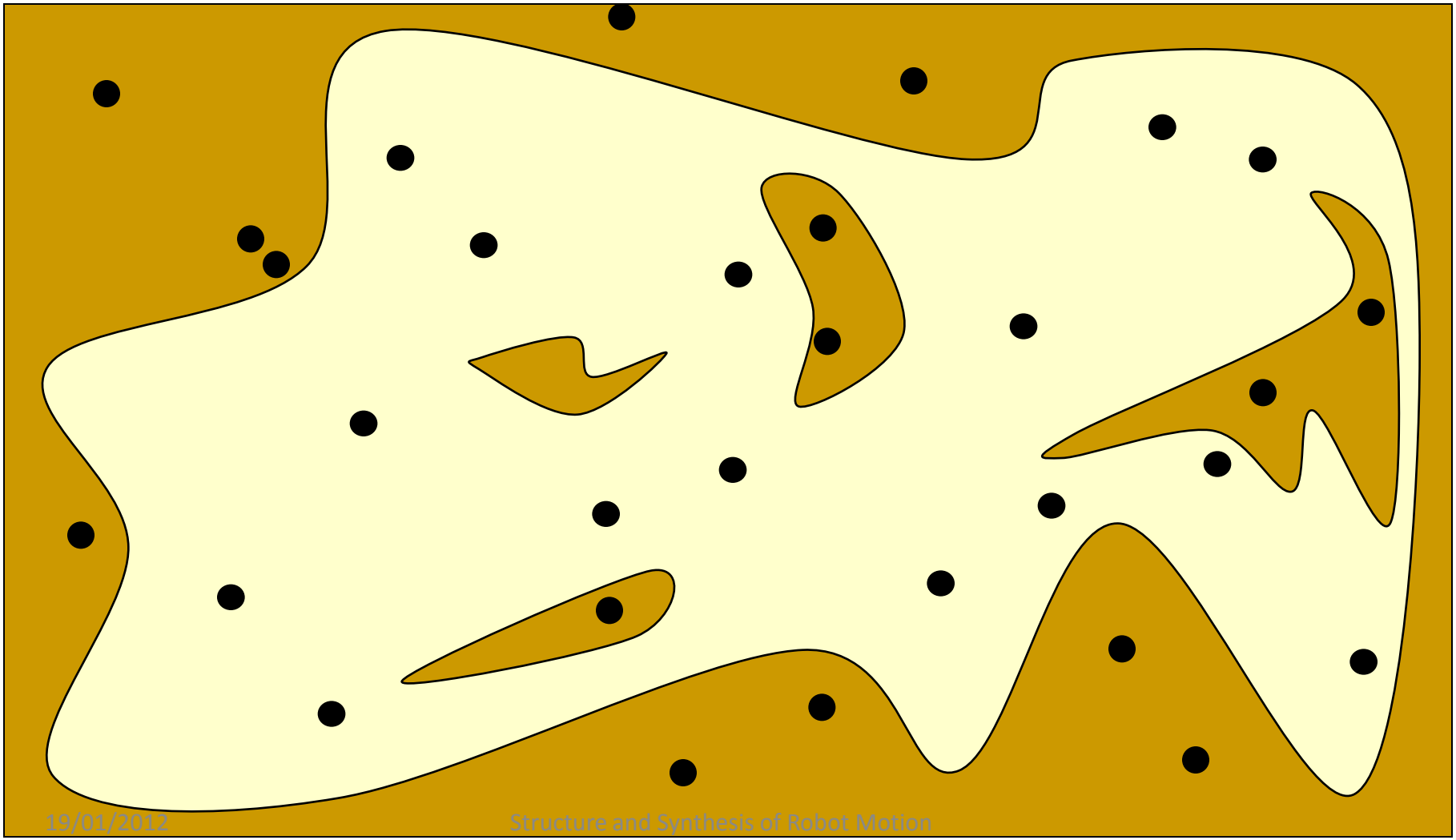
Probabilistic Roadmap: Step 1

Configurations are sampled by picking coordinates at random



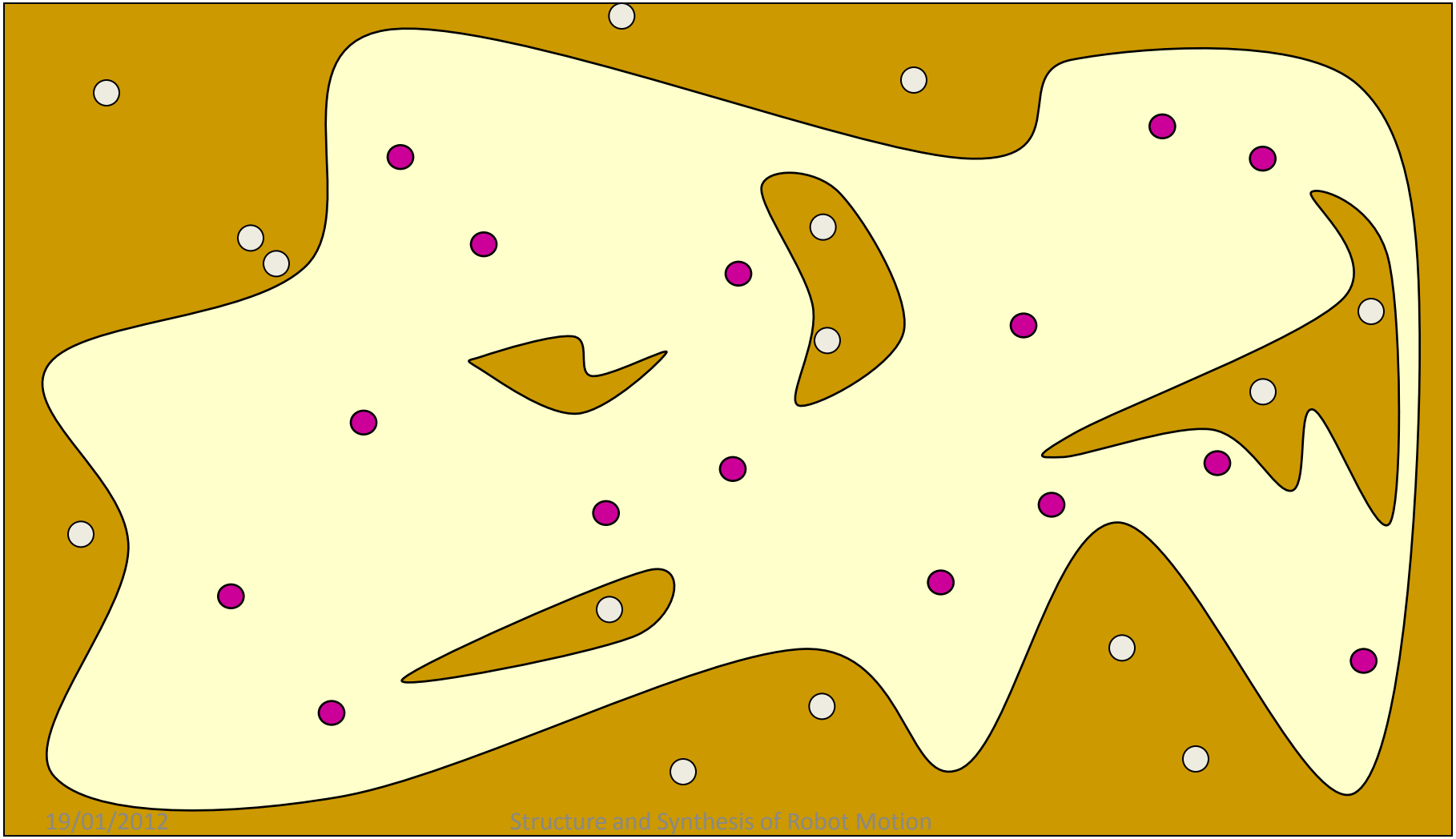
Probabilistic Roadmap: Step 1

Configurations are sampled by picking coordinates at random



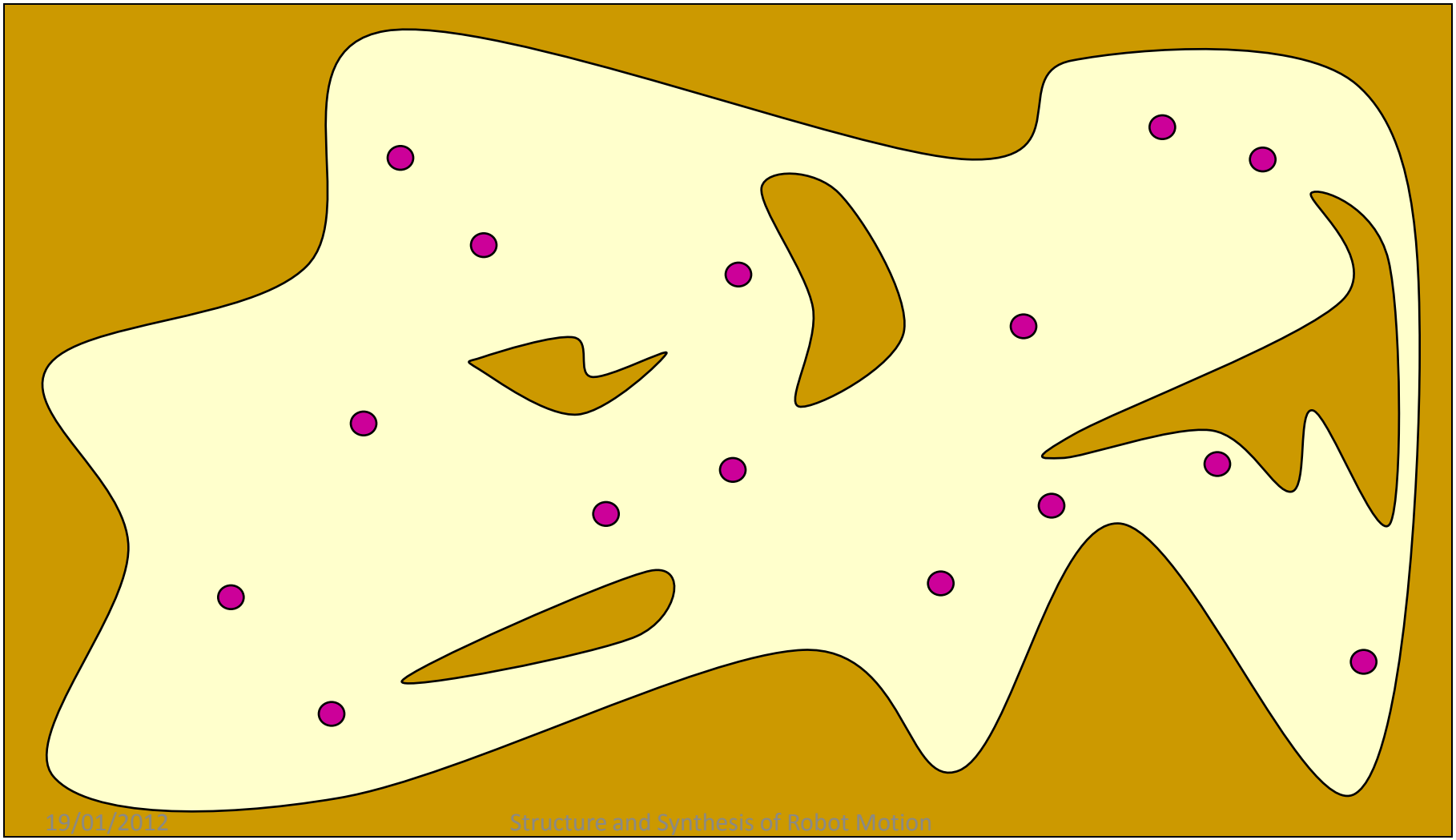
Probabilistic Roadmap: Step 2

Sampled configurations are tested for collision (in workspace!)



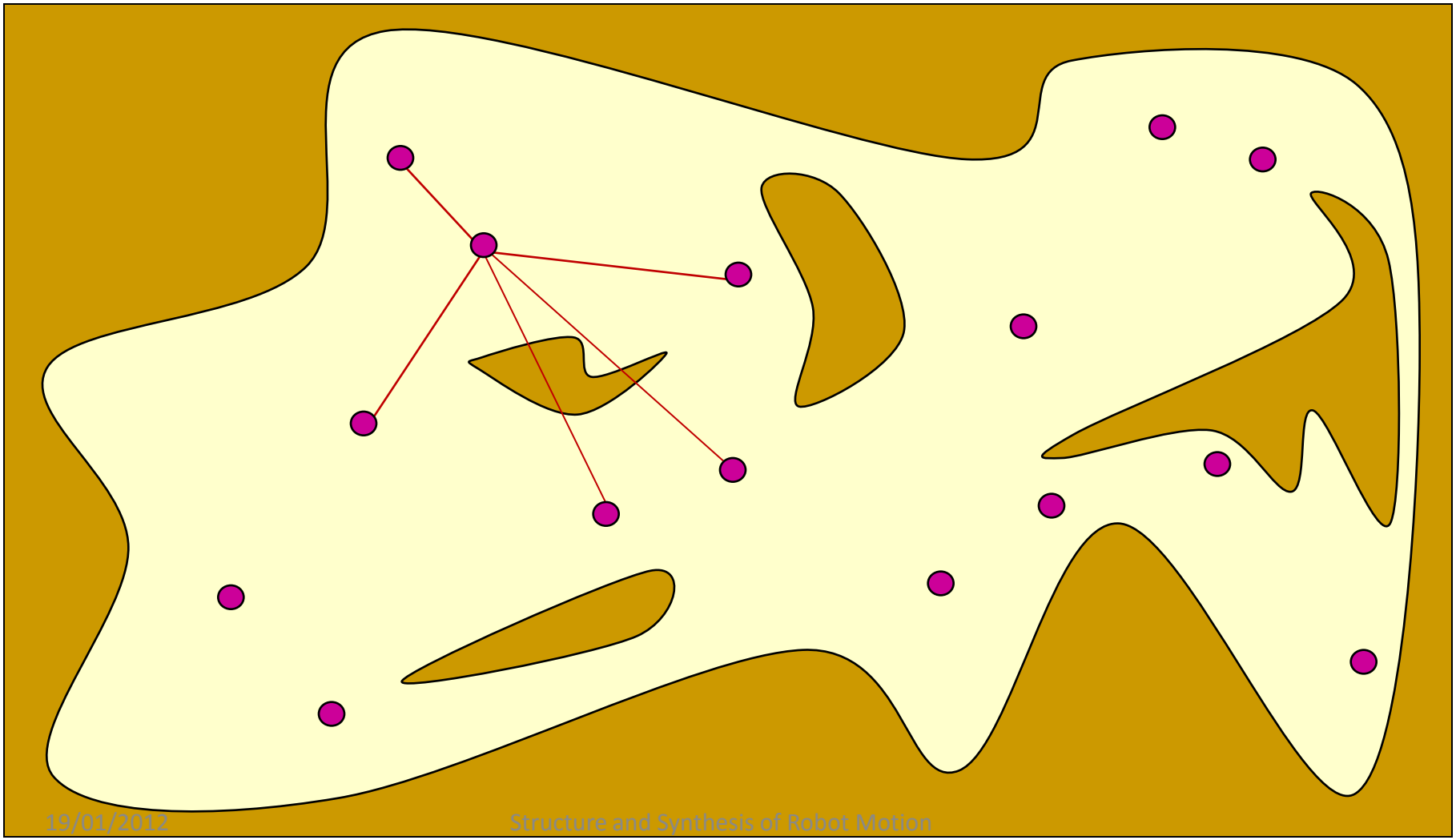
Probabilistic Roadmap: Step 3

The collision-free configurations are retained as “milestones”



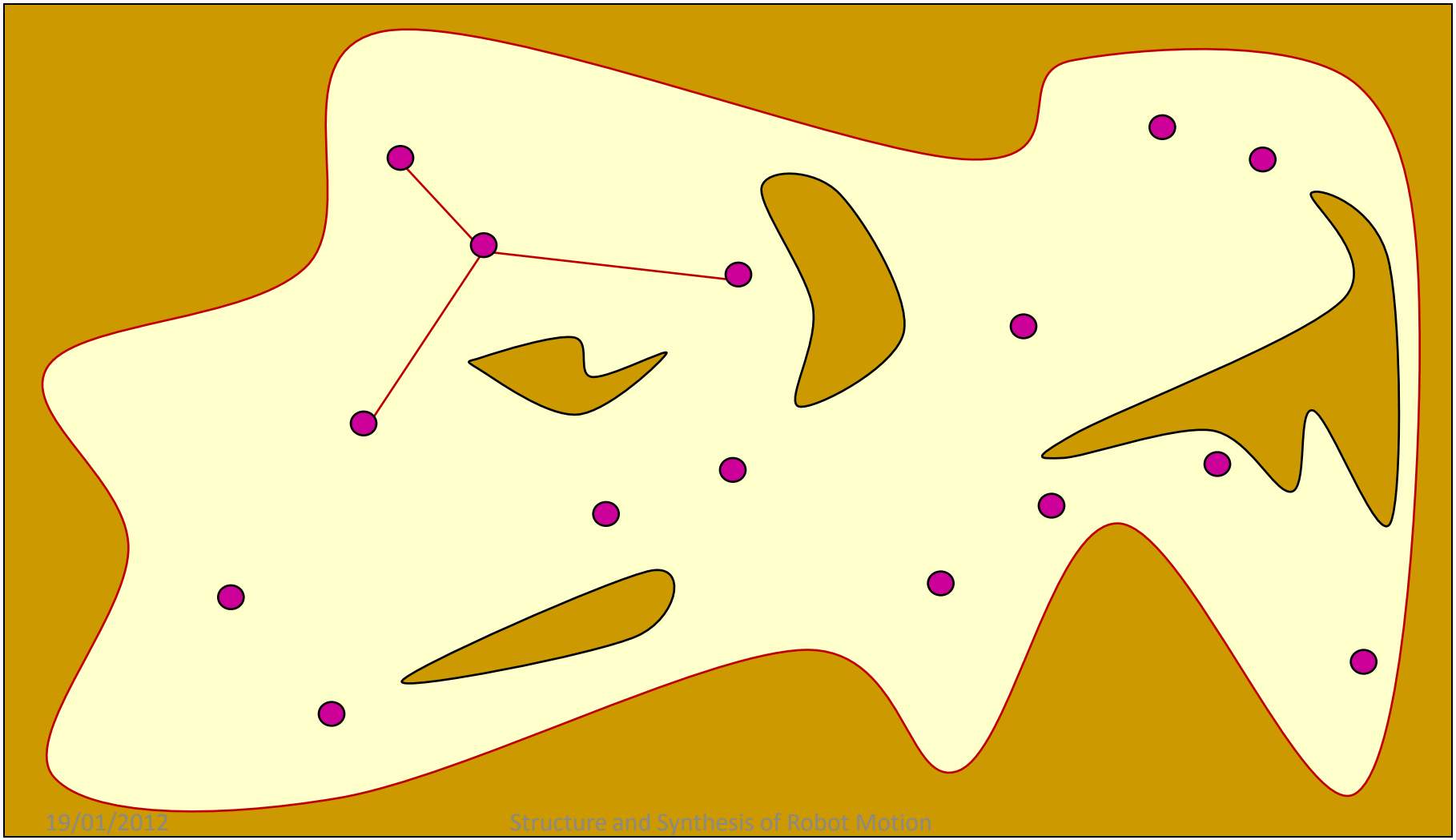
Probabilistic Roadmap: Step 4

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



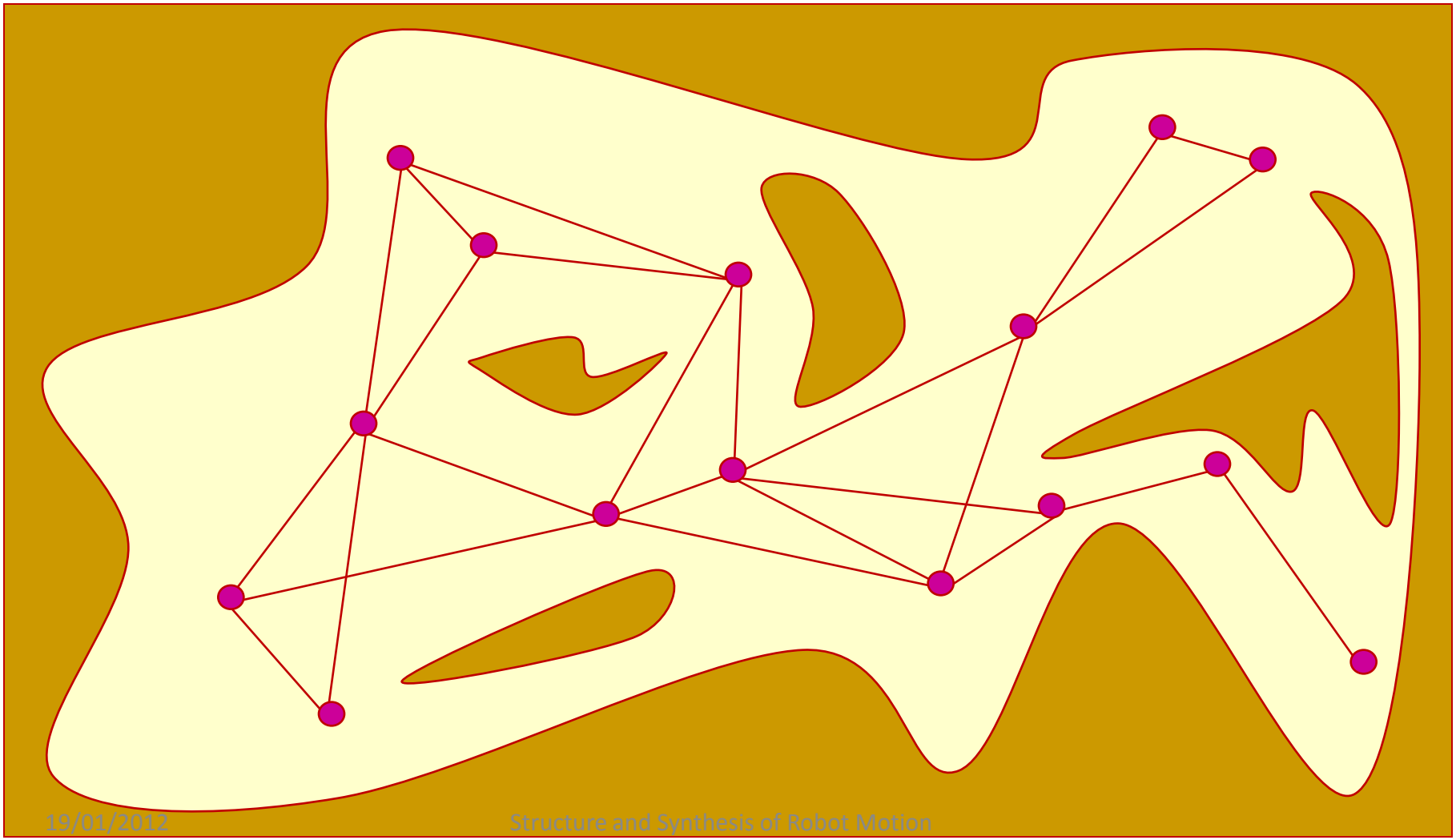
Probabilistic Roadmap: Step 5

Retain only collision-free links



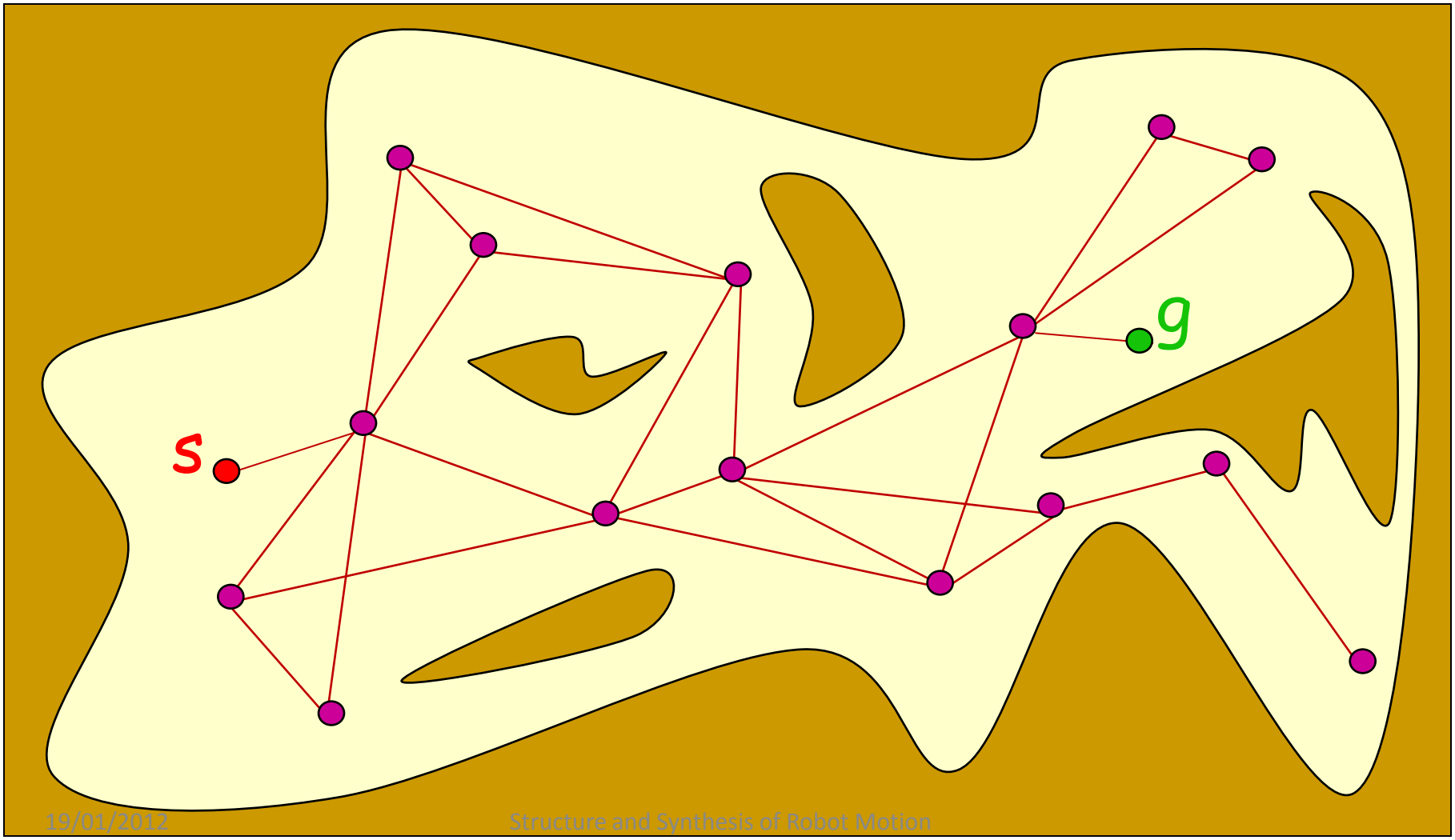
Probabilistic Roadmap: Step 6

The collision-free links are used to form the PRM



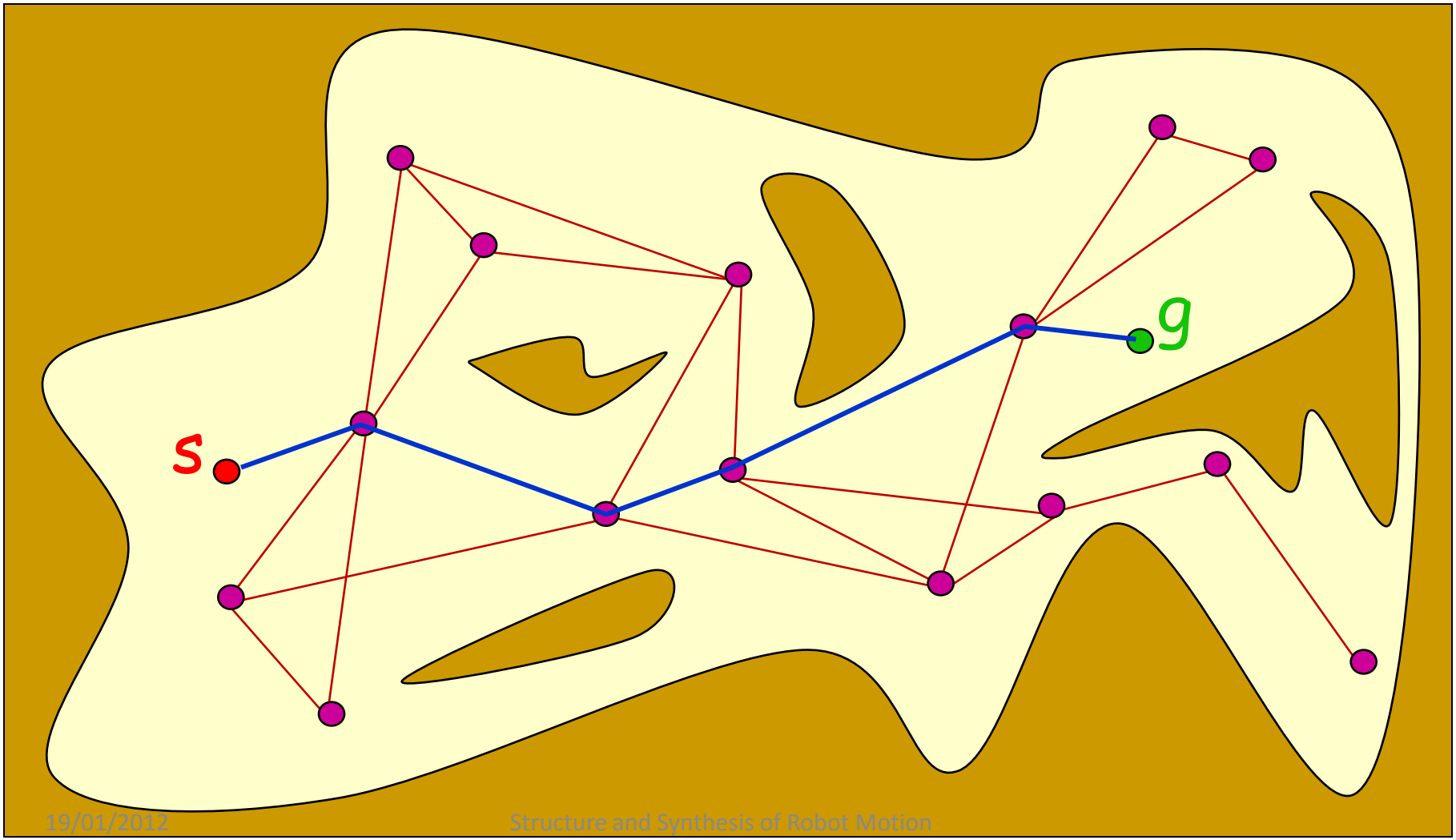
Probabilistic Roadmap: Query Time

The start and goal configurations are included as milestones



Probabilistic Roadmap: Finding the Path

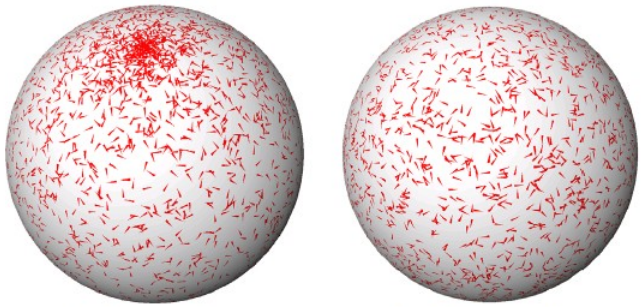
Search PRM (e.g., A*) for a path from s to g



In Practice, Key Requirements for PRMs

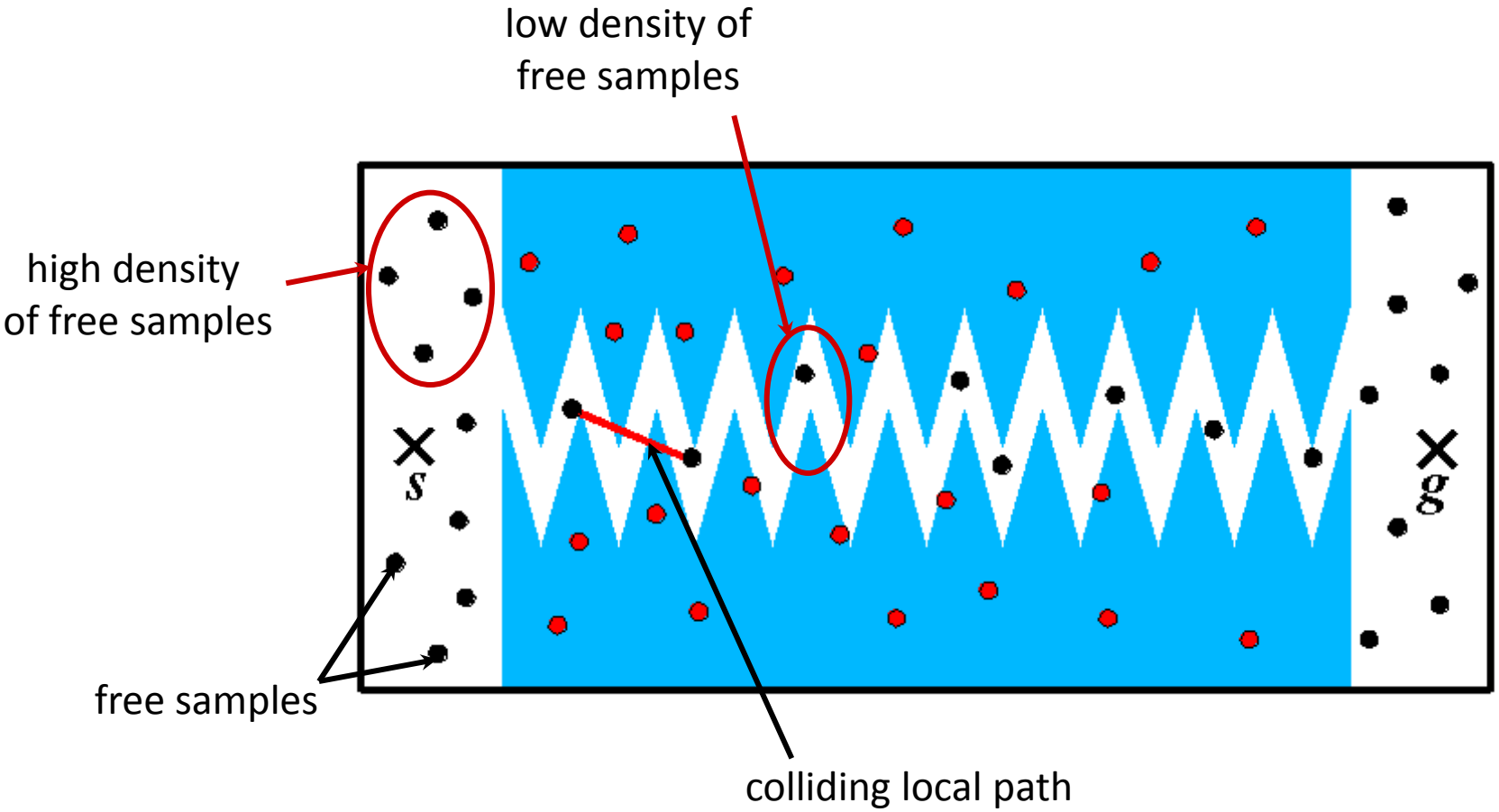
PRMs need the following procedures to be extremely efficient:

- Collision Detection (numerous libraries exist)
  - To check if a sampled configuration is in free space
  - To check if local plan is feasible
- Nearest-neighbour search (e.g., kd-trees)
  - To pick target nodes for any given sample node
- Sampling: different strategies give you different benefits



(a) Naïve sampling (b) Uniform sampling

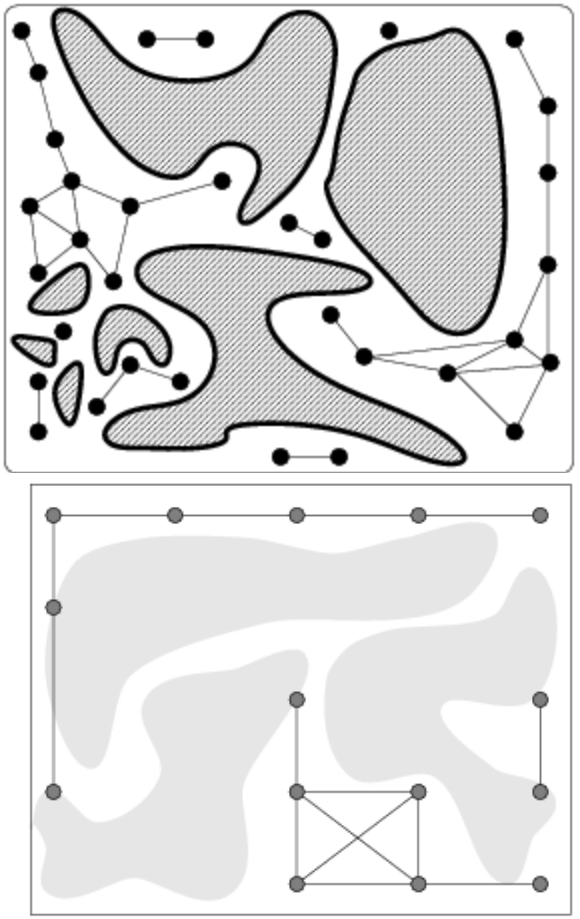
What can Go Wrong? Narrow Passages



What can Go Wrong? Lack of Connectivity

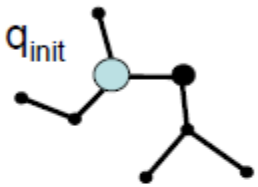
A combination of issues:

- Collection of narrow passages can impede roadmap growth
- Non-uniformity of a finite set of samples may lead to clustering – so *k*-NN connections may not cover the whole space
- Similarly, boundary points are sparsely sampled



Single-Query Planning: Rapidly Exploring Random Trees (RRT)

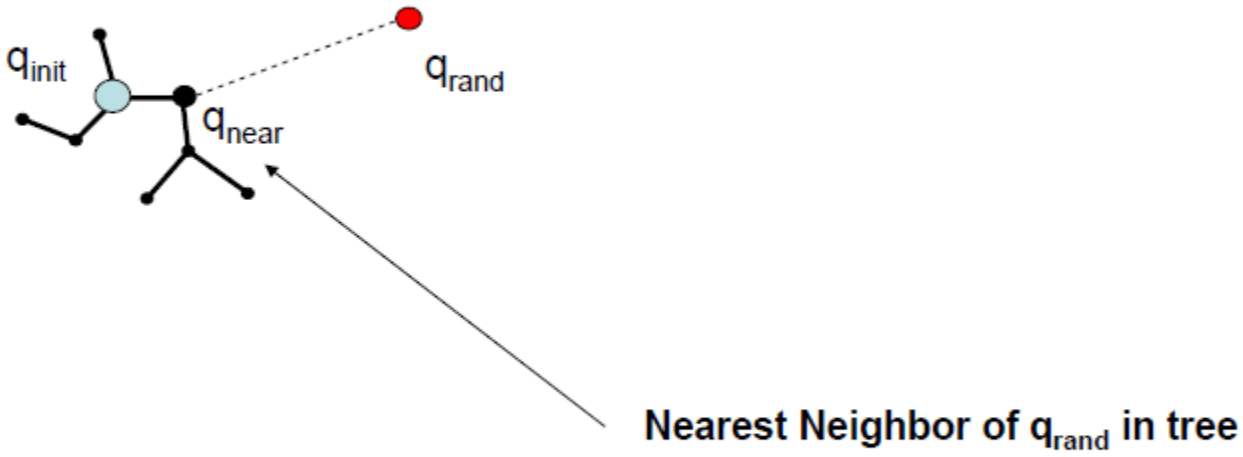
[LaValle '98, LaValle & Kuffner '00]



Use a tree that explores space,
much like an unfurling search tree

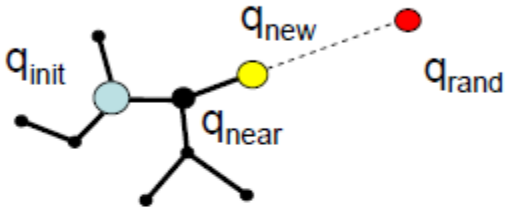
RRT: Step 1

[LaValle '98, LaValle & Kuffner '00]



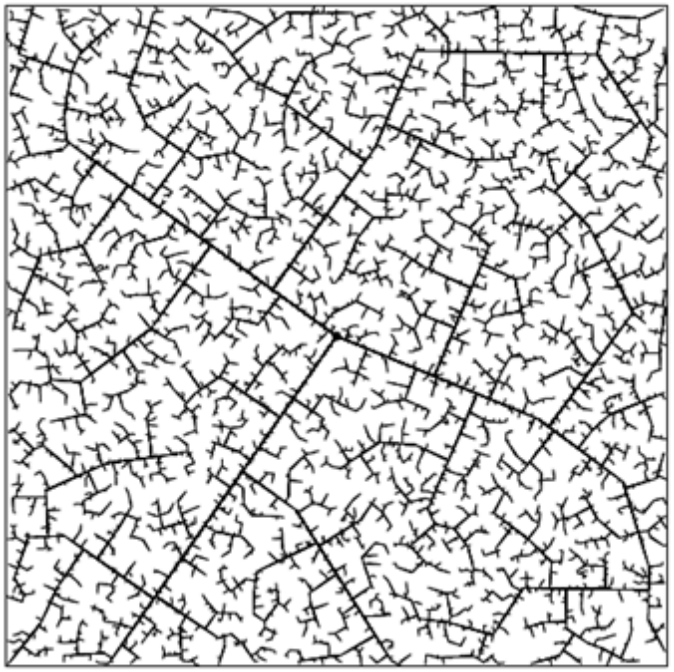
RRT: Step 2

[LaValle '98, LaValle & Kuffner '00]



RRT @ Work

45 iterations

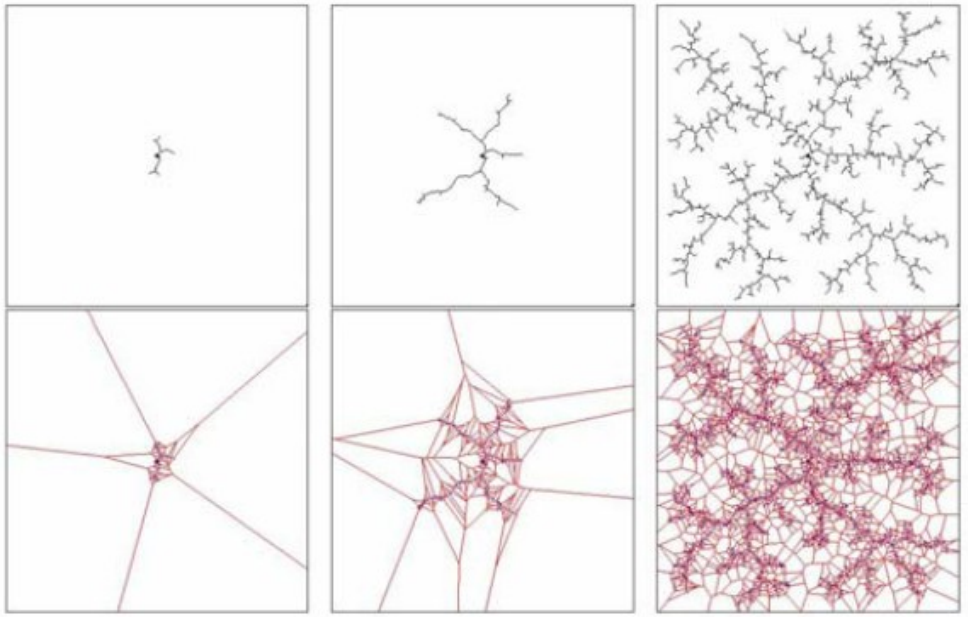


2345 iterations

In the early iterations, the RRT quickly reaches the unexplored parts. However, the RRT is dense in the limit (with probability one), which means that it gets arbitrarily to any point in the space.

What is Really Going On?

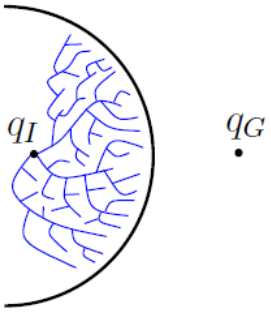
Incrementally, a partition in terms of a Voronoi diagram is refined.



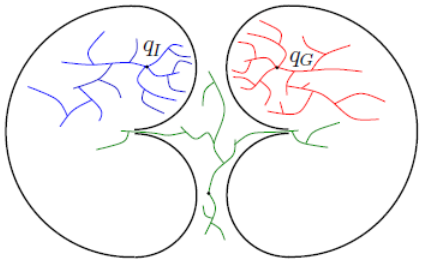
The probability that a path is found increases exponentially with the number of iterations.

[Kuffner & LaValle '00]

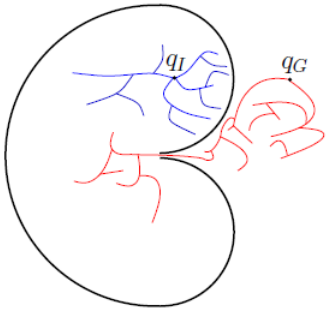
In Practice, Many Extensions may be Needed



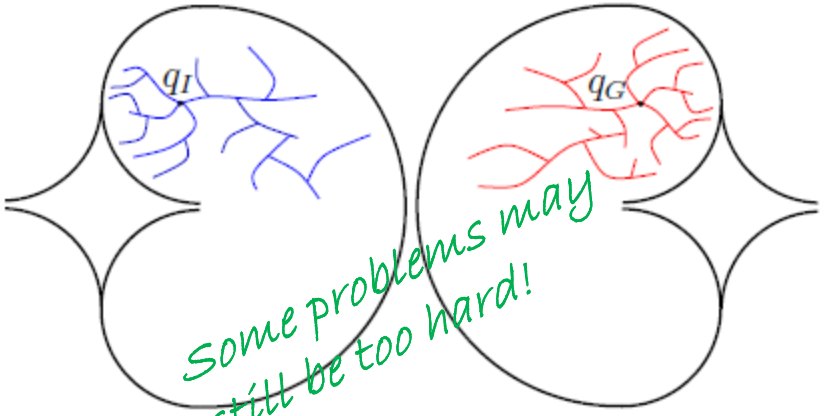
Multi-resolution exploration



Multi-directional search



Bi-directional search



Acknowledgments

Many pictures/slides are adapted from the following sources:

- Steve LaValle, *Planning Algorithms* (Ch. 5)
- J-C. Latombe, Stanford University, lecture slides
- M. Stilman, Georgia Tech, lecture slides