Planification de mouvement de manipulation

Florent Lamiraux

CNRS-LAAS, Toulouse, France

Planification de mouvement

Planification de mouvement de manipulation

Definitions

A manipulation motion

- is the motion of
  - one or several robots and of
  - one or several objects
- such that each object
  - either is in a stable position, or
  - is moved by one or several robots.

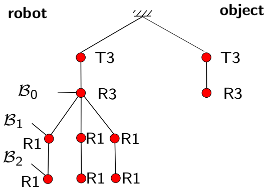
Definitions

A manipulation motion

- is the motion of
  - one or several robots and of
  - one or several objects
- such that each object
  - either is in a stable position, or
  - is moved by one or several robots.

Composite robot

Kinematic chain composed of each robot and of each object



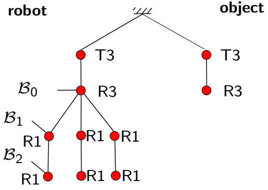
$$\mathbf{q} = (q_0, \cdots, q_{n_r}, q_{n_r+1}, \cdots, q_{n_r+n_o})$$

The configuration space of a composite robot is the cartesian product of the configuration spaces of each robot and object.

$$\mathcal{C} = \mathcal{C}_{r1} \times \mathcal{C}_{r_{nb\ robots}} \times SE(3)^{nb\ objets}$$

Composite robot

Kinematic chain composed of each robot and of each object



$$\mathbf{q} = (q_0, \cdots, q_{n_r}, q_{n_r+1}, \cdots, q_{n_r+n_o})$$

The configuration space of a composite robot is the cartesian product of the configuration spaces of each robot and object.

$$\mathcal{C} = \mathcal{C}_{r1} \times \mathcal{C}_{r_{nb\ robots}} \times SE(3)^{nb\ objets}$$

Numerical constraints

Constraints to which manipulation motions are subject can be expressed numerically.

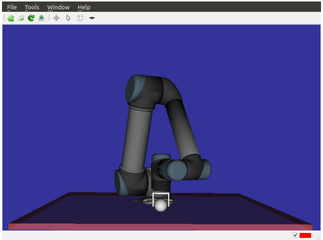
- Numerical constraints :

$$f(\mathbf{q}) = 0, \qquad \begin{array}{l} m \in \mathbb{N}, \\ f \in C^1(\mathcal{C}, \mathbb{R}^m) \end{array}$$

- Parameterizable numerical constraints :

$$f(\mathbf{q}) = f_0, \qquad \begin{array}{l} m \in \mathbb{N}, \\ f \in C^1(\mathcal{C}, \mathbb{R}^m) \\ f_0 \in \mathbb{R}^m \end{array}$$

Example : robot manipulating a ball



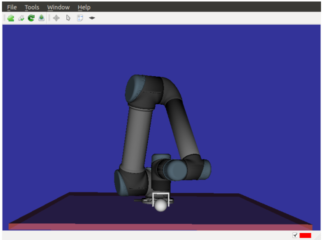
$$\mathcal{C} = [-\pi, \pi]^6 \times \mathbb{R}^3 \quad (1)$$

$$\mathbf{q} = (q_0, \cdots, q_5, x_b, y_b, z_b) \quad (2)$$

Two *states* :

- `placement` : the ball is lying on the table,
- `grasp` : the ball is hold by the end-effector.

Example : robot manipulating a ball



Each state is defined by a numerical constraint

- placement

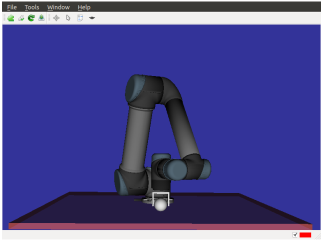
$$z_b = 0$$

- grasp

$$\mathbf{x}_{gripper}(q_0, \cdots, q_5) - \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = 0$$

Each state is a sub-manifold of the configuration space

Example : robot manipulating a ball



Each state is defined by a numerical constraint

- `placement`

$$z_b = 0$$

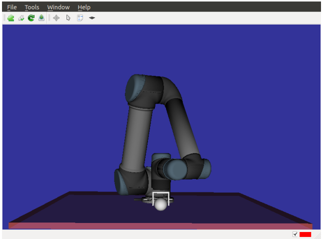
- `grasp`

$$\mathbf{x}_{gripper}(q_0, \cdots, q_5) - \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = 0$$

Each state is a sub-manifold of the configuration space

Example : robot manipulating a ball

Motion constraints

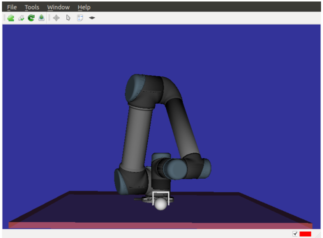


Two *types of motion* :

- `transit` : the ball is lying and **fixed** on the table,
- `transfer` : the ball moves with the end-effector.

Example : robot manipulating a ball

Motion constraints



- `transit`

$$\left.\begin{array}{rcl} x_b & = & x_0 \\ y_b & = & y_0 \end{array}\right\} \quad \text{parameterizable}$$

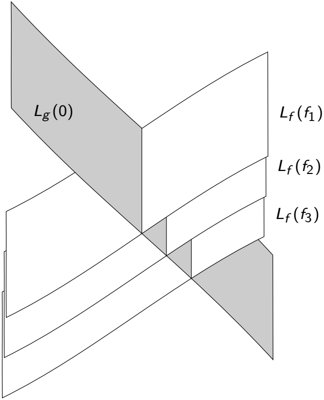
$$\left. z_b = 0 \right\} \quad \text{simple}$$

- `transfer`

$$\mathbf{x}_{gripper}(q_0, \cdots, q_5) - \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = 0$$

Foliation

Motion constraints define a foliation of the admissible configuration space (`grasp` ∪ `placement`).



- f : position of the ball

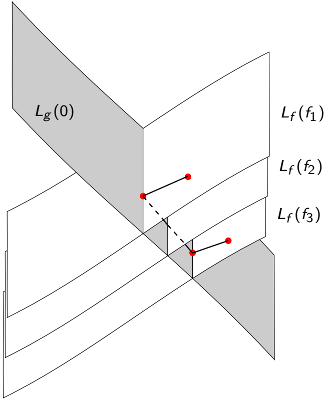
$$L_f(f_1) = \{\mathbf{q} \in \mathcal{C}, f(\mathbf{q}) = f_1\}$$

- g : grasp of the ball

$$L_g(0) = \{\mathbf{q} \in \mathcal{C}, g(\mathbf{q}) = 0\}$$

Foliation

Motion constraints define a foliation of the admissible configuration space (`grasp` ∪ `placement`).



Solution to a manipulation planning problem is a concatenation of *transit* and *transfer* paths.

General case

In a manipulation problem,

- the state of the system is subject to
  - numerical constraints
- system trajectories are subject to
  - numerical constraints
  - parameterizable numerical constraints.

General case

In a manipulation problem,

- the state of the system is subject to
  - numerical constraints
- system trajectories are subject to
  - numerical constraints
  - parameterizable numerical constraints.

General case

In a manipulation problem,

- the state of the system is subject to
  - numerical constraints
- system trajectories are subject to
  - ~~numerical constraints~~
  - parameterizable numerical constraints, the dimension of the parameter being possibly less than the dimension of the constraint.
  - parameter value is constant along trajectories.

General case

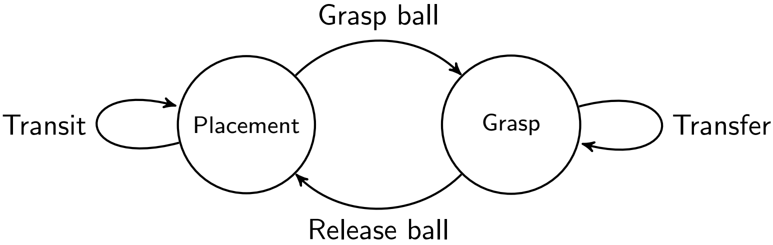
In a manipulation problem,

- the state of the system is subject to
  - numerical constraints
- system trajectories are subject to
  - ~~numerical constraints~~
  - parameterizable numerical constraints, the dimension of the parameter being possibly less than the dimension of the constraint.
  - parameter value is constant along trajectories.

Constraint graph

A manipulation planning problem can be represented by a *manipulation graph*.

- **Nodes** or *states* are numerical constraints.
- **Edges** or *transitions* are parameterizable numerical constraints.



Projecting configuration on constraint

- $\mathbf{q}_0$ configuration,
- $f(\mathbf{q}) = 0$ non-linear constraint,
- ϵ numerical tolerance

Projection $(\mathbf{q}_0, f)$:

$\mathbf{q} = \mathbf{q}_0$; $\alpha = 0.95$

for i from 1 to max_iter :

$$\mathbf{q} = \mathbf{q} - \alpha \left(\frac{\partial f}{\partial \mathbf{q}}(\mathbf{q})\right)^{+} f(\mathbf{q})$$

if $\|f(\mathbf{q})\| < \epsilon$: return $\mathbf{q}$

return failure

Projecting path on constraint

- *path* : mapping from $[0, 1]$ to $\mathcal{C}$
- $f(\mathbf{q}) = 0$ non-linear constraint,
- ϵ numerical tolerance

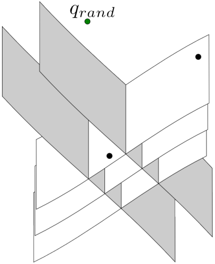
Projection (*path*, f) :

  if $\|f(path(0))\| > \epsilon$ or $\|f(path(1))\| > \epsilon$:

    return failure

Algorithm

Manipulation RRT



Manipulation RRT

$\mathbf{q}_{rand}$ = shoot_random_config()

for each connected component :

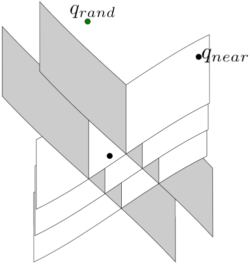
  $\mathbf{q}_{near}$ = nearest_neighbour($\mathbf{q}_{rand}$, *roadmap*)
  e = select_transition($\mathbf{q}_{near}$)
  $\mathbf{q}_{proj}$ = generate_target_config($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$, e)
  $\mathbf{q}_{new}$ = extend($\mathbf{q}_{near}$, $\mathbf{q}_{proj}$, edge)
  roadmap.insert_node($\mathbf{q}_{new}$)
  roadmap.insert_edge(e, $\mathbf{q}_{near}$, $\mathbf{q}_{new}$)
  new_nodes.append ($\mathbf{q}_{new}$)

for $(\mathbf{q}_0, \mathbf{q}_1) \in$ pairs$(\mathbf{q}^1_{new}, ..., \mathbf{q}^{n_{cc}}_{new})$:

  connect (roadmap, $\mathbf{q}_0$, $\mathbf{q}_1$)

Algorithm

Manipulation RRT

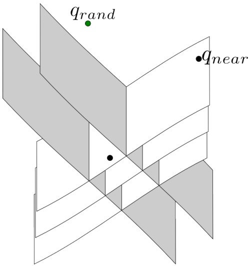


Manipulation RRT

$\mathbf{q}_{rand}$ = shoot_random_config()
for each connected component :
    $\mathbf{q}_{near}$ = nearest_neighbour($\mathbf{q}_{rand}$, *roadmap*)
    e = select_transition($\mathbf{q}_{near}$)
    $\mathbf{q}_{proj}$ = generate_target_config($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$, e)
    $\mathbf{q}_{new}$ = extend($\mathbf{q}_{near}$, $\mathbf{q}_{proj}$, edge)
    roadmap.insert_node($\mathbf{q}_{new}$)
    roadmap.insert_edge(e, $\mathbf{q}_{near}$, $\mathbf{q}_{new}$)
    new_nodes.append ($\mathbf{q}_{new}$)
for $(\mathbf{q}_0, \mathbf{q}_1) \in$ pairs$(\mathbf{q}^1_{new}, \ldots, \mathbf{q}^{n_{cc}}_{new})$:
    connect (roadmap, $\mathbf{q}_0$, $\mathbf{q}_1$)

Algorithm

Manipulation RRT



Manipulation RRT

$\mathbf{q}_{rand}$ = shoot_random_config()

for each connected component :

$\mathbf{q}_{near}$ = nearest_neighbour($\mathbf{q}_{rand}$, *roadmap*)

e = select_transition($\mathbf{q}_{near}$)

$\mathbf{q}_{proj}$ = generate_target_config($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$, e)

$\mathbf{q}_{new}$ = extend($\mathbf{q}_{near}$, $\mathbf{q}_{proj}$, edge)

roadmap.insert_node($\mathbf{q}_{new}$)

roadmap.insert_edge(e, $\mathbf{q}_{near}$, $\mathbf{q}_{new}$)

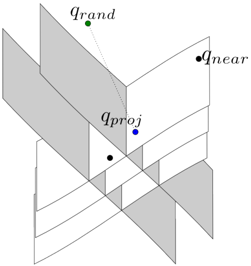
new_nodes.append ($\mathbf{q}_{new}$)

for $(\mathbf{q}_0, \mathbf{q}_1) \in$ pairs$(\mathbf{q}_{new}^1, \ldots, \mathbf{q}_{new}^{n_{cc}})$:

connect (roadmap, $\mathbf{q}_0$, $\mathbf{q}_1$)

Algorithm

Manipulation RRT

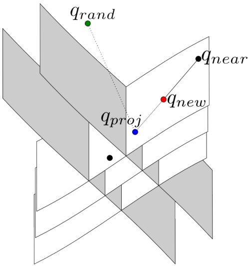


Manipulation RRT

$\mathbf{q}_{rand}$ = shoot_random_config()
for each connected component :
 $\mathbf{q}_{near}$ = nearest_neighbour($\mathbf{q}_{rand}$, *roadmap*)
 e = select_transition($\mathbf{q}_{near}$)
 $\mathbf{q}_{proj}$ = generate_target_config($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$, e)
 $\mathbf{q}_{new}$ = extend($\mathbf{q}_{near}$, $\mathbf{q}_{proj}$, edge)
 roadmap.insert_node($\mathbf{q}_{new}$)
 roadmap.insert_edge(e, $\mathbf{q}_{near}$, $\mathbf{q}_{new}$)
 new_nodes.append ($\mathbf{q}_{new}$)
for $(\mathbf{q}_0, \mathbf{q}_1) \in$ pairs($\mathbf{q}^1_{new}, \ldots, \mathbf{q}^{n_{cc}}_{new}$) :
 connect (roadmap, $\mathbf{q}_0$, $\mathbf{q}_1$)

Algorithm

Manipulation RRT

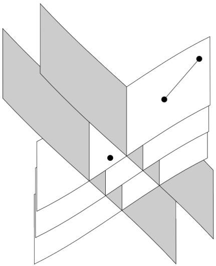


Manipulation RRT

$\mathbf{q}_{rand}$ = shoot_random_config()
for each connected component :
    $\mathbf{q}_{near}$ = nearest_neighbour($\mathbf{q}_{rand}$, *roadmap*)
    e = select_transition($\mathbf{q}_{near}$)
    $\mathbf{q}_{proj}$ = generate_target_config($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$, e)
    $\mathbf{q}_{new}$ = extend($\mathbf{q}_{near}$, $\mathbf{q}_{proj}$, edge)
    roadmap.insert_node($\mathbf{q}_{new}$)
    roadmap.insert_edge(e, $\mathbf{q}_{near}$, $\mathbf{q}_{new}$)
    new_nodes.append ($\mathbf{q}_{new}$)
for $(\mathbf{q}_0, \mathbf{q}_1) \in$ pairs($\mathbf{q}^1_{new}, \ldots, \mathbf{q}^{n_{cc}}_{new}$) :
    connect (roadmap, $\mathbf{q}_0$, $\mathbf{q}_1$)

Algorithm

Manipulation RRT



Manipulation RRT

$\mathbf{q}_{rand} = \text{shoot\_random\_config}()$
for each connected component :

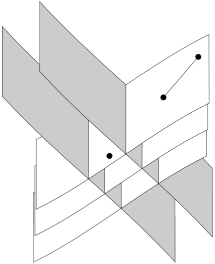
$\mathbf{q}_{near} = \text{nearest\_neighbour}(\mathbf{q}_{rand}, \textit{roadmap})$
$e = \text{select\_transition}(\mathbf{q}_{near})$
$\mathbf{q}_{proj} = \text{generate\_target\_config}(\mathbf{q}_{near}, \mathbf{q}_{rand}, e)$
$\mathbf{q}_{new} = \text{extend}(\mathbf{q}_{near}, \mathbf{q}_{proj}, \text{edge})$
roadmap.insert_node($\mathbf{q}_{new}$)
roadmap.insert_edge(e, $\mathbf{q}_{near}$, $\mathbf{q}_{new}$)
new_nodes.append ($\mathbf{q}_{new}$)

for $(\mathbf{q}_0, \mathbf{q}_1) \in \text{pairs}(\mathbf{q}^1_{new}, \ldots, \mathbf{q}^{n_{cc}}_{new})$:

connect (roadmap, $\mathbf{q}_0, \mathbf{q}_1$)

Algorithm

Manipulation RRT



Manipulation RRT

$\mathbf{q}_{rand} = \text{shoot\_random\_config}()$

for each connected component :

$\mathbf{q}_{near} = \text{nearest\_neighbour}(\mathbf{q}_{rand}, \textit{roadmap})$

$e = \text{select\_transition}(\mathbf{q}_{near})$

$\mathbf{q}_{proj} = \text{generate\_target\_config}(\mathbf{q}_{near}, \mathbf{q}_{rand}, e)$

$\mathbf{q}_{new} = \text{extend}(\mathbf{q}_{near}, \mathbf{q}_{proj}, \text{edge})$

$\textit{roadmap}.\text{insert\_node}(\mathbf{q}_{new})$

$\textit{roadmap}.\text{insert\_edge}(\text{e}, \mathbf{q}_{near}, \mathbf{q}_{new})$

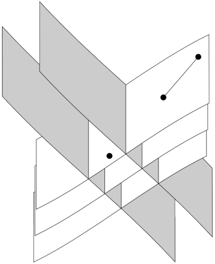
$\text{new\_nodes.append}\ (\mathbf{q}_{new})$

for $(\mathbf{q}_0, \mathbf{q}_1) \in \text{pairs}(\mathbf{q}^1_{new}, \ldots, \mathbf{q}^{n_{cc}}_{new})$:

connect (roadmap, $\mathbf{q}_0, \mathbf{q}_1$)

Algorithm

Manipulation RRT



Manipulation RRT

$\mathbf{q}_{rand}$ = shoot_random_config()

for each connected component :

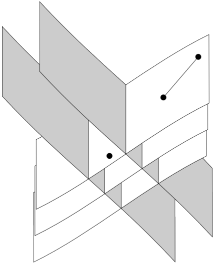
- $\mathbf{q}_{near}$ = nearest_neighbour($\mathbf{q}_{rand}$, *roadmap*)
- e = select_transition($\mathbf{q}_{near}$)
- $\mathbf{q}_{proj}$ = generate_target_config($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$, e)
- $\mathbf{q}_{new}$ = extend($\mathbf{q}_{near}$, $\mathbf{q}_{proj}$, edge)
- *roadmap*.insert_node($\mathbf{q}_{new}$)
- *roadmap*.insert_edge(e, $\mathbf{q}_{near}$, $\mathbf{q}_{new}$)
- new_nodes.append ($\mathbf{q}_{new}$)

for $(\mathbf{q}_0, \mathbf{q}_1) \in$ pairs($\mathbf{q}^1_{new}, \ldots, \mathbf{q}^{n_{cc}}_{new}$) :

- connect (roadmap, $\mathbf{q}_0, \mathbf{q}_1$)

Algorithm

Manipulation RRT



Manipulation RRT

$\mathbf{q}_{rand} = \text{shoot\_random\_config}()$

for each connected component :

$\mathbf{q}_{near} = \text{nearest\_neighbour}(\mathbf{q}_{rand}, \textit{roadmap})$

$e = \text{select\_transition}(\mathbf{q}_{near})$

$\mathbf{q}_{proj} = \text{generate\_target\_config}(\mathbf{q}_{near}, \mathbf{q}_{rand}, e)$

$\mathbf{q}_{new} = \text{extend}(\mathbf{q}_{near}, \mathbf{q}_{proj}, \text{edge})$

$\textit{roadmap}.\text{insert\_node}(\mathbf{q}_{new})$

$\textit{roadmap}.\text{insert\_edge}(\text{e}, \mathbf{q}_{near}, \mathbf{q}_{new})$

$\text{new\_nodes.append}\ (\mathbf{q}_{new})$

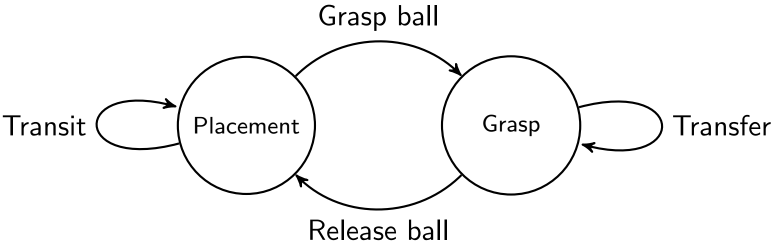
for $(\mathbf{q}_0, \mathbf{q}_1) \in \text{pairs}(\mathbf{q}^1_{new}, \ldots, \mathbf{q}^{n_{cc}}_{new})$:

connect (roadmap, $\mathbf{q}_0, \mathbf{q}_1$)

Select transition

$e = \text{select\_transition}(\mathbf{q}_{near})$

Outward edges of each node are given a probability distribution. The transition from a node to another node is chosen by random sampling.



Generate target configuration

$\mathbf{q}_{proj} = \text{generate\_target\_config}(\mathbf{q}_{near}, \mathbf{q}_{rand}, e)$

Once edge e has been selected, $\mathbf{q}_{rand}$ is *projected* onto the destination node n_{dest} in a configuration reachable by $\mathbf{q}_{near}$.

$$\begin{aligned} f_e(\mathbf{q}_{proj}) &= f_e(\mathbf{q}_{near}) \\ f_{dest}(\mathbf{q}_{proj}) &= 0 \end{aligned}$$

Extend

$$\mathbf{q}_{new} = \text{extend}(\mathbf{q}_{near}, \mathbf{q}_{proj}, \text{edge})$$

Project straight path $[\mathbf{q}_{near}, \mathbf{q}_{proj}]$ on edge constraint :

- if projection successful and projected path collision free

$$\mathbf{q}_{new} \leftarrow \mathbf{q}_{proj}$$

- otherwise $(\mathbf{q}_{near}, \mathbf{q}_{new}) \leftarrow$ largest path interval tested as collision-free with successful projection.

$$\forall \mathbf{q} \in (\mathbf{q}_{near}, \mathbf{q}_{new}), \; f_e(\mathbf{q}) = f_e(\mathbf{q}_{near})$$

Extend

$$\mathbf{q}_{new} = \text{extend}(\mathbf{q}_{near}, \mathbf{q}_{proj}, \text{edge})$$

Project straight path $[\mathbf{q}_{near}, \mathbf{q}_{proj}]$ on edge constraint :

- if projection successful and projected path collision free

$$\mathbf{q}_{new} \leftarrow \mathbf{q}_{proj}$$

- otherwise $(\mathbf{q}_{near}, \mathbf{q}_{new}) \leftarrow$ largest path interval tested as collision-free with successful projection.

$$\forall \mathbf{q} \in (\mathbf{q}_{near}, \mathbf{q}_{new}), \ f_e(\mathbf{q}) = f_e(\mathbf{q}_{near})$$

Extend

$$\mathbf{q}_{new} = \text{extend}(\mathbf{q}_{near}, \mathbf{q}_{proj}, \text{edge})$$

Project straight path $[\mathbf{q}_{near}, \mathbf{q}_{proj}]$ on edge constraint :

- if projection successful and projected path collision free

$$\mathbf{q}_{new} \leftarrow \mathbf{q}_{proj}$$

- otherwise $(\mathbf{q}_{near}, \mathbf{q}_{new}) \leftarrow$ largest path interval tested as collision-free with successful projection.

$$\forall \mathbf{q} \in (\mathbf{q}_{near}, \mathbf{q}_{new}),\ f_e(\mathbf{q}) = f_e(\mathbf{q}_{near})$$

Connect

connect (roadmap, $\mathbf{q}_0, \mathbf{q}_1$) :

s_0 = state ($\mathbf{q}_0$)

s_1 = state ($\mathbf{q}_1$)

e = transition (n_0, n_1)

if e and $f_e(\mathbf{q}_0) == f_e(\mathbf{q}_1)$:

if p = projected_path (e, $\mathbf{q}_0$, $\mathbf{q}_1$) collision-free :

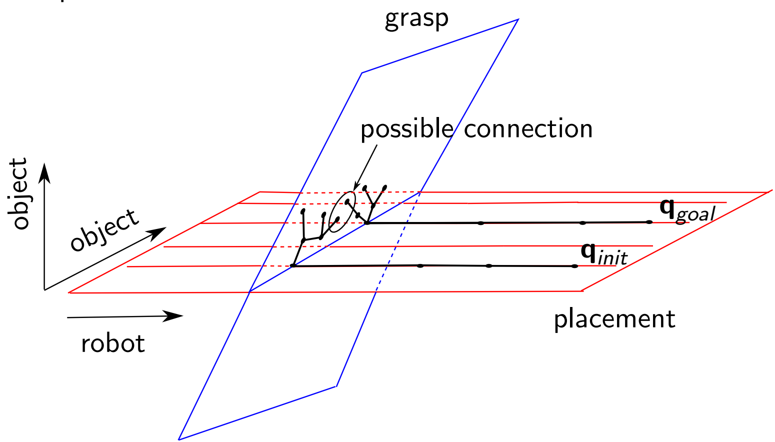
roadmap.insert_edge (e, $\mathbf{q}_0$,$\mathbf{q}_1$)

return

Connecting trees

Manipulation RRT is initialized with $\mathbf{q}_{init}$, $\mathbf{q}_{goal}$.

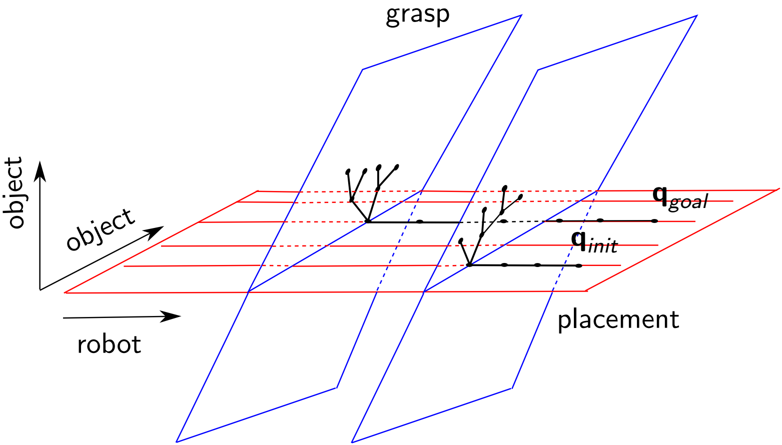
- 2 connected components.
- possible connection.



Connecting trees : general case

Manipulation RRT is initialized with $\mathbf{q}_{init}$, $\mathbf{q}_{goal}$.

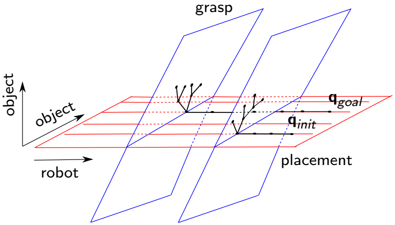
- 2 connected components,
- no possible connection.

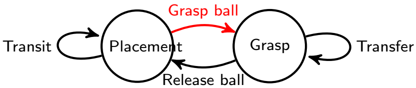


Connecting trees : general case

Manipulation RRT is initialized with $\mathbf{q}_{init}$, $\mathbf{q}_{goal}$.

- 2 connected components,
- no possible connection.





Relative positions as numerical constraints

Let $T_1 = T_{(R_1,t_1)}$ and $T_2 = T_{(R_2,t_2)}$ be two rigid-body transformations. The relative transformation $T_{2/1} = T_1^{-1} \circ T_2$ can be represented by a vector of dimension 6 :

$$\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}$$

where

$$\mathbf{u} = R_1^T (t_2 - t_1)$$

$R_1^T R_2$ is the matrix of the rotation around axis $\mathbf{v}/\|\mathbf{v}\|$ and of angles $\|\mathbf{v}\|$.

Solving non-linear constraints

Problem : find $\mathbf{q} \in \mathcal{C}$ such that

$$f(\mathbf{q}) = 0$$

given an initial value $\mathbf{q}_0 \in \mathcal{C}$