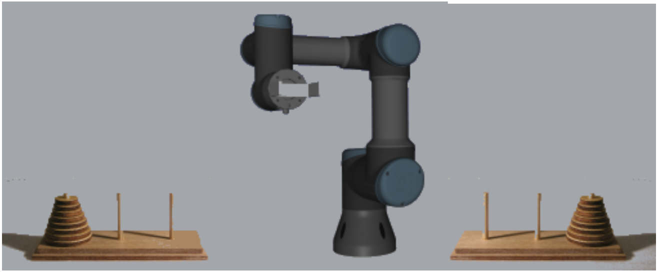
Motion planning

Florent Lamiraux

CNRS-LAAS, Toulouse, France

Task and Motion Planning

Definition

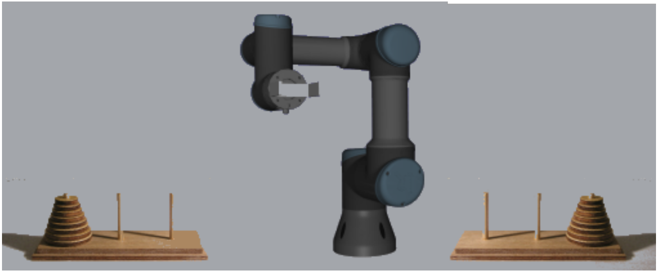


Given

- One or several robots,
- One or several objects,
- initial configurations for robots and objects
- goal configurations for robots and objects

Task and Motion planning : automatically computing a feasible trajectory between the initial and goal configurations.

Definition

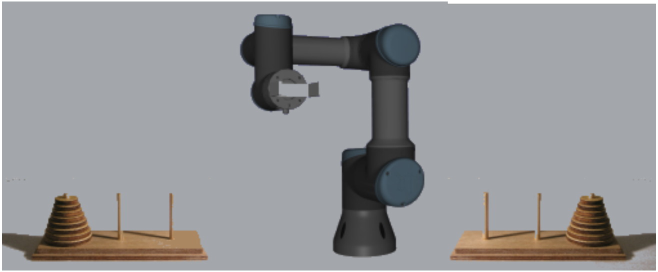


Given

- One or several robots,
- One or several objects,
- initial configurations for robots and objects
- goal configurations for robots and objects

Task and Motion planning : automatically computing a feasible trajectory between the initial and goal configurations.

Definition

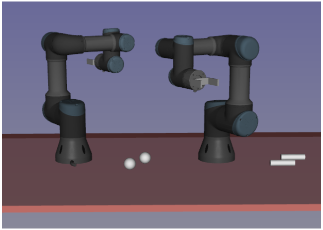


Given

- One or several robots,
- One or several objects,
- initial configurations for robots and objects
- goal configurations for robots and objects

Task and Motion planning : automatically computing a feasible trajectory between the initial and goal configurations.

Configuration space



Configuration

$$\mathbf{q} = (\mathbf{q}_{r_1}, \mathbf{q}_{r_2}, \mathbf{q}_{c_1}, \mathbf{q}_{c_2}, \mathbf{q}_{s_1}, \mathbf{q}_{s_2})$$
$$\mathbf{q}_{r_1}, \mathbf{q}_{r_2} \in \mathbb{R}^6$$
$$\mathbf{q}_{c_1}, \mathbf{q}_{c_2}, \mathbf{q}_{s_1}, \mathbf{q}_{s_2} \in \mathbb{R}^7$$

where

$\mathbf{q}_{r_1} = (q1, \cdots, q_6)$ is the vector of joint angles,

$\mathbf{q}_{c_1} = (x, y, z, X, Y, Z, W)$

$W + Xi + Yj + Zk$is a **unit** quaternion.

The configuration space is a differential manifold.

Quaternions

Non-commutative field isomorphic to $\mathbb{R}^4$, spanned by three elements i, j, k that satisfy the following relations :

$$i^2 = j^2 = k^2 = ijk = -1$$

from which we immediately deduce

$$ij = k, \; jk = i, \; ki = j$$

Hamilton (1843)

Quaternions

Non-commutative field isomorphic to $\mathbb{R}^4$, spanned by three elements i, j, k that satisfy the following relations :

$$i^2 = j^2 = k^2 = ijk = -1$$

from which we immediately deduce

$$ij = k,\ \ jk = i,\ \ ki = j$$

Hamilton (1843)

Unit Quaternions and rotations

Let $q = q_0 + q_1 i + q_2 j + q_3 k$ be a unit quaternion :

$$q_0^2 + q_3^2 + q_2^2 + q_3^2 = 1$$

$\forall x = (x_0, x_1, x_2) \in \mathbb{R}^3$, let $u = x_0 i + x_1 j + x_2 k$

$$q \,.\, u \,.\, q^* = y_0 i + y_1 j + y_2 k$$

where $q^* = q_0 - q_1 i - q_2 j - q_3 k$ is the conjugate of q.
$y = (y_0, y_1, y_2)$ is the image of x by the rotation of matrix

$$\begin{pmatrix} 1 - 2(q_2^2 + q_3^2) & 2q_2 q_1 - 2q_3 q_0 & 2q_3 q_1 + 2q_2 q_0 \\ 2q_2 q_1 + 2q_3 q_0 & 1 - 2(q_1^2 + q_3^2) & 2q_3 q_2 - 2q_1 q_0 \\ 2q_3 q_1 - 2q_2 q_0 & 2q_3 q_2 + 2q_1 q_0 & 1 - 2(q_1^2 + q_2^2) \end{pmatrix}$$

Unit Quaternions and rotations

Let $q = q_0 + q_1 i + q_2 j + q_3 k$ be a unit quaternion :

$$q_0^2 + q_3^2 + q_2^2 + q_3^2 = 1$$

$\forall x = (x_0, x_1, x_2) \in \mathbb{R}^3$, let $u = x_0 i + x_1 j + x_2 k$

$$q \,.\, u \,.\, q^* = y_0 i + y_1 j + y_2 k$$

where $q^* = q_0 - q_1 i - q_2 j - q_3 k$ is the conjugate of q.
$y = (y_0, y_1, y_2)$ is the image of x by the rotation of matrix

$$\begin{pmatrix} 1 - 2(q_2^2 + q_3^2) & 2q_2q_1 - 2q_3q_0 & 2q_3q_1 + 2q_2q_0 \\ 2q_2q_1 + 2q_3q_0 & 1 - 2(q_1^2 + q_3^2) & 2q_3q_2 - 2q_1q_0 \\ 2q_3q_1 - 2q_2q_0 & 2q_3q_2 + 2q_1q_0 & 1 - 2(q_1^2 + q_2^2) \end{pmatrix}$$

Unit Quaternions and rotations

Let $q = q_0 + q_1 i + q_2 j + q_3 k$ be a unit quaternion :

$$q_0^2 + q_3^2 + q_2^2 + q_3^2 = 1$$

$\forall x = (x_0, x_1, x_2) \in \mathbb{R}^3$, let $u = x_0 i + x_1 j + x_2 k$

$$q \,.\, u \,.\, q^* = y_0 i + y_1 j + y_2 k$$

where $q^* = q_0 - q_1 i - q_2 j - q_3 k$ is the conjugate of q.
$y = (y_0, y_1, y_2)$ is the image of x by the rotation of matrix

$$\begin{pmatrix} 1 - 2(q_2^2 + q_3^2) & 2q_2 q_1 - 2q_3 q_0 & 2q_3 q_1 + 2q_2 q_0 \\ 2q_2 q_1 + 2q_3 q_0 & 1 - 2(q_1^2 + q_3^2) & 2q_3 q_2 - 2q_1 q_0 \\ 2q_3 q_1 - 2q_2 q_0 & 2q_3 q_2 + 2q_1 q_0 & 1 - 2(q_1^2 + q_2^2) \end{pmatrix}$$

Unit Quaternions and rotations

- Notice that q and $-q$ represent the same rotation

Definitions

- Workspace : $\mathcal{W} = \mathbb{R}^2$ or $\mathbb{R}^3$: space in which the robot evolves
- Obstacle in workspace : compact subset of $\mathcal{W}$, denoted by $\mathcal{O}$.
- Configuration space : $\mathcal{C}$.
- Position in configuration $\mathbf{q}$ of a point $M \in \mathcal{B}_i$: $\mathbf{x}_i(M, \mathbf{q})$.
- Obstacle in the configuration space :

$$\begin{aligned} \mathcal{C}_{obst} = \{\mathbf{q} \in \mathcal{C}, \quad & \exists i \in \{1, \cdots, m\},\ \exists M \in \mathcal{B}_i,\ \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ or} \\ & \exists i, j \in \{1, \cdots, m\},\ \exists M_i \in \mathcal{B}_i,\ \exists M_j \in \mathcal{B}_j, \\ & \mathbf{x}_i(M_i, \mathbf{q}) = x_j(M_j, \mathbf{q})\} \end{aligned}$$

- Free configuration space : $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$.

Definitions

- Workspace : $\mathcal{W} = \mathbb{R}^2$ or $\mathbb{R}^3$: space in which the robot evolves
- Obstacle in workspace : compact subset of $\mathcal{W}$, denoted by $\mathcal{O}$.
- Configuration space : $\mathcal{C}$.
- Position in configuration $\mathbf{q}$ of a point $M \in \mathcal{B}_i$: $\mathbf{x}_i(M, \mathbf{q})$.
- Obstacle in the configuration space :

$$\begin{aligned}\mathcal{C}_{obst} = \{\mathbf{q} \in \mathcal{C}, \quad & \exists i \in \{1, \cdots, m\},\ \exists M \in \mathcal{B}_i,\ \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ or} \\ & \exists i, j \in \{1, \cdots, m\},\ \exists M_i \in \mathcal{B}_i,\ \exists M_j \in \mathcal{B}_j, \\ & \mathbf{x}_i(M_i, \mathbf{q}) = x_j(M_j, \mathbf{q})\}\end{aligned}$$

- Free configuration space : $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$.

Definitions

- Workspace : $\mathcal{W} = \mathbb{R}^2$ or $\mathbb{R}^3$: space in which the robot evolves
- Obstacle in workspace : compact subset of $\mathcal{W}$, denoted by $\mathcal{O}$.
- Configuration space : $\mathcal{C}$.
- Position in configuration $\mathbf{q}$ of a point $M \in \mathcal{B}_i$: $\mathbf{x}_i(M, \mathbf{q})$.
- Obstacle in the configuration space :

$$\begin{aligned}\mathcal{C}_{obst} = \{\mathbf{q} \in \mathcal{C}, \; & \exists i \in \{1, \cdots, m\}, \; \exists M \in \mathcal{B}_i, \; \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ or} \\ & \exists i, j \in \{1, \cdots, m\}, \; \exists M_i \in \mathcal{B}_i, \; \exists M_j \in \mathcal{B}_j, \\ & \mathbf{x}_i(M_i, \mathbf{q}) = x_j(M_j, \mathbf{q})\}\end{aligned}$$

- Free configuration space : $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$.

Definitions

- Workspace : $\mathcal{W} = \mathbb{R}^2$ or $\mathbb{R}^3$: space in which the robot evolves
- Obstacle in workspace : compact subset of $\mathcal{W}$, denoted by $\mathcal{O}$.
- Configuration space : $\mathcal{C}$.
- Position in configuration $\mathbf{q}$ of a point $M \in \mathcal{B}_i$: $\mathbf{x}_i(M, \mathbf{q})$.
- Obstacle in the configuration space :

$$\begin{aligned} \mathcal{C}_{obst} = \{\mathbf{q} \in \mathcal{C}, \quad & \exists i \in \{1, \cdots, m\},\ \exists M \in \mathcal{B}_i,\ \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ or} \\ & \exists i, j \in \{1, \cdots, m\},\ \exists M_i \in \mathcal{B}_i,\ \exists M_j \in \mathcal{B}_j, \\ & \mathbf{x}_i(M_i, \mathbf{q}) = x_j(M_j, \mathbf{q})\} \end{aligned}$$

- Free configuration space : $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$.

Definitions

- Workspace : $\mathcal{W} = \mathbb{R}^2$ or $\mathbb{R}^3$: space in which the robot evolves
- Obstacle in workspace : compact subset of $\mathcal{W}$, denoted by $\mathcal{O}$.
- Configuration space : $\mathcal{C}$.
- Position in configuration $\mathbf{q}$ of a point $M \in \mathcal{B}_i$: $\mathbf{x}_i(M, \mathbf{q})$.
- Obstacle in the configuration space :

$$
\begin{aligned}
\mathcal{C}_{obst} = \{\mathbf{q} \in \mathcal{C}, \quad & \exists i \in \{1, \cdots, m\},\ \exists M \in \mathcal{B}_i,\ \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ or} \\
& \exists i, j \in \{1, \cdots, m\},\ \exists M_i \in \mathcal{B}_i,\ \exists M_j \in \mathcal{B}_j, \\
& \mathbf{x}_i(M_i, \mathbf{q}) = x_j(M_j, \mathbf{q})\}
\end{aligned}
$$

- Free configuration space : $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$.

Definitions

- Workspace : $\mathcal{W} = \mathbb{R}^2$ or $\mathbb{R}^3$: space in which the robot evolves
- Obstacle in workspace : compact subset of $\mathcal{W}$, denoted by $\mathcal{O}$.
- Configuration space : $\mathcal{C}$.
- Position in configuration $\mathbf{q}$ of a point $M \in \mathcal{B}_i$: $\mathbf{x}_i(M, \mathbf{q})$.
- Obstacle in the configuration space :

$$\begin{aligned} \mathcal{C}_{obst} \quad = \{\mathbf{q} \in \mathcal{C}, \quad & \exists i \in \{1, \cdots, m\},\ \exists M \in \mathcal{B}_i,\ \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ or} \\ & \exists i, j \in \{1, \cdots, m\},\ \exists M_i \in \mathcal{B}_i,\ \exists M_j \in \mathcal{B}_j, \\ & \mathbf{x}_i(M_i, \mathbf{q}) = x_j(M_j, \mathbf{q})\} \end{aligned}$$

- Free configuration space : $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$.

Motion

- Configuration space :
  - differential manifold
- Motion :
  - continuous function from $[0, 1]$ to $\mathcal{C}$.
- Collision-free motion :
  - continuous function from $[0, 1]$ to $\mathcal{C}_{free}$.

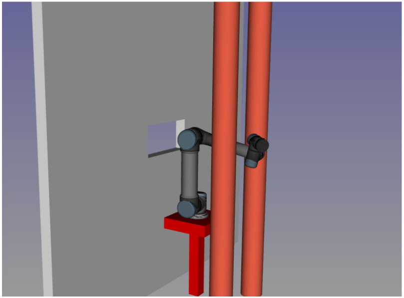
Motion

- Configuration space :
  - differential manifold
- Motion :
  - continuous function from $[0, 1]$ to $\mathcal{C}$.
- Collision-free motion :
  - continuous function from $[0, 1]$ to $\mathcal{C}_{free}$.

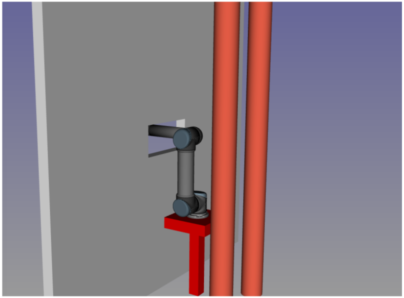
Motion

- Configuration space :
  - differential manifold
- Motion :
  - continuous function from $[0, 1]$ to $\mathcal{C}$.
- Collision-free motion :
  - continuous function from $[0, 1]$ to $\mathcal{C}_{free}$.

Motion planning problem



initial configuration



goal configuration

$$\mathcal{C} = \mathbb{R}^6$$

History

- Before the 1990's : mainly a mathematical problem
  - real algebraic geometry,
  - decidability : Schwartz and Sharir 1982,
    - Tarski theorem, Collins decomposition,
- from the 1990's : an algorithmic problem
  - random sampling (1993),
  - asymptotically optimal random sampling (2011).

History

- Before the 1990's : mainly a mathematical problem
  - real algebraic geometry,
  - decidability : Schwartz and Sharir 1982,
    - Tarski theorem, Collins decomposition,
- from the 1990's : an algorithmic problem
  - random sampling (1993),
  - asymptotically optimal random sampling (2011).

Random methods

- In the early 1990's, random methods started being developed
- Principle
  - shoot random configurations
  - test whether they are in collision
  - build a graph (roadmap) the nodes of which are free configurations
  - and the edges of which are collision-free linear interpolations

Random methods

- In the early 1990's, random methods started being developed
- Principle
  - shoot random configurations
  - test whether they are in collision
  - build a graph (roadmap) the nodes of which are free configurations
  - and the edges of which are collision-free linear interpolations

Random methods

- In the early 1990's, random methods started being developed
- Principle
  - shoot random configurations
  - test whether they are in collision
  - build a graph (roadmap) the nodes of which are free configurations
  - and the edges of which are collision-free linear interpolations

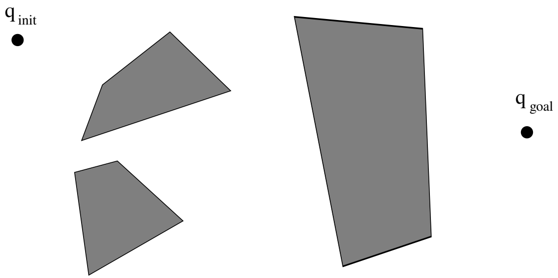
Random methods

- In the early 1990's, random methods started being developed
- Principle
  - shoot random configurations
  - test whether they are in collision
  - build a graph (roadmap) the nodes of which are free configurations
  - and the edges of which are collision-free linear interpolations

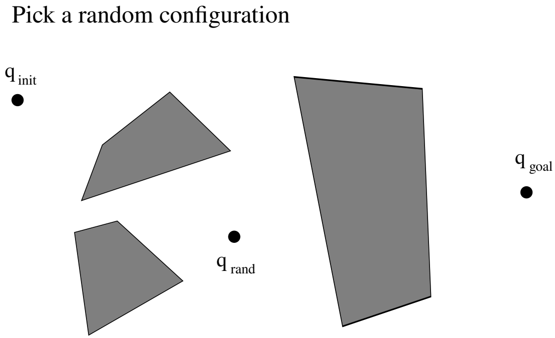
Random methods

- In the early 1990's, random methods started being developed
- Principle
  - shoot random configurations
  - test whether they are in collision
  - build a graph (roadmap) the nodes of which are free configurations
  - and the edges of which are collision-free linear interpolations

Rapidly exploring Random Tree (RRT) 2000

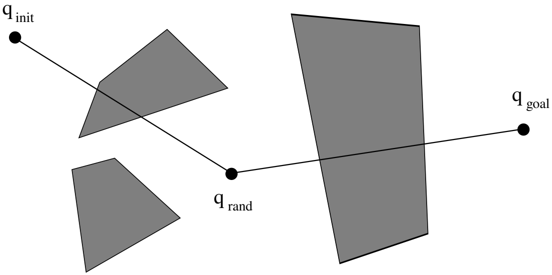


Rapidly exploring Random Tree (RRT) 2000



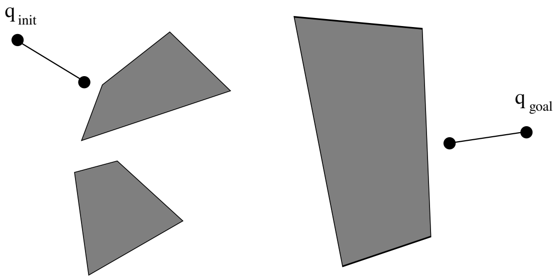
Rapidly exploring Random Tree (RRT) 2000

Try to connect it to the nearest nodes of each connected component



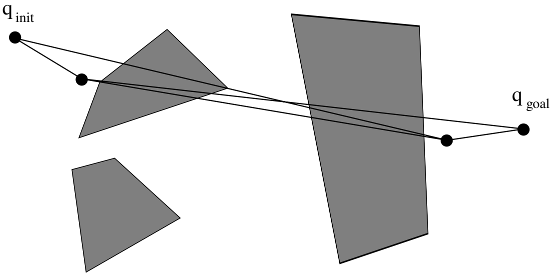
Rapidly exploring Random Tree (RRT) 2000

Keep collision-free parts of paths

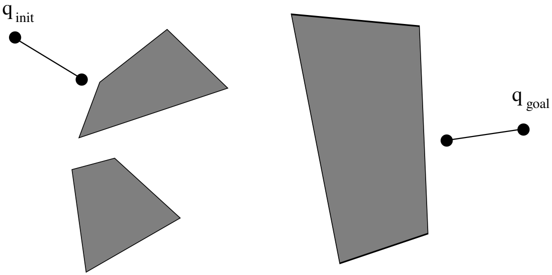


Rapidly exploring Random Tree (RRT) 2000

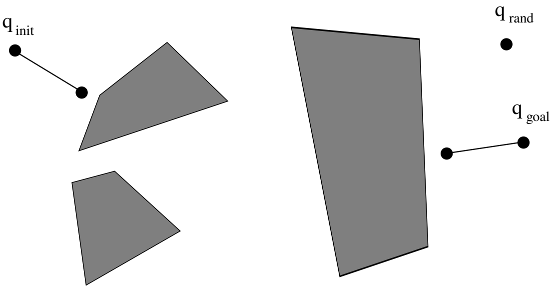
Try to connect new nodes to nearest nodes of other connected components



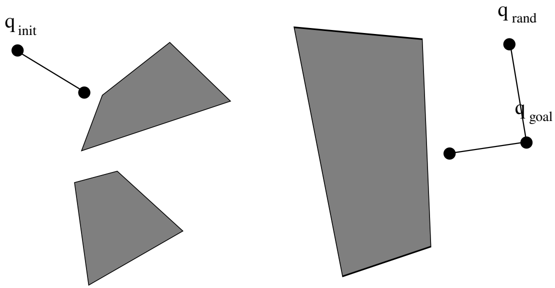
Rapidly exploring Random Tree (RRT) 2000



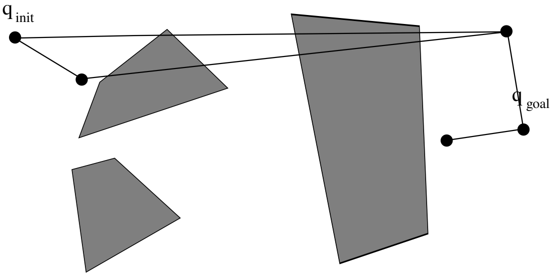
Rapidly exploring Random Tree (RRT) 2000



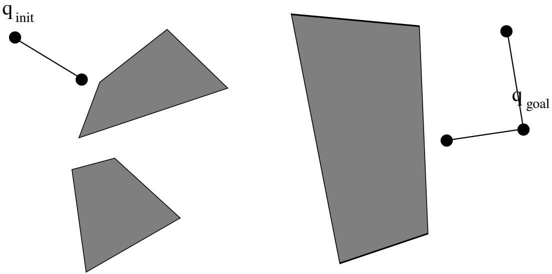
Rapidly exploring Random Tree (RRT) 2000



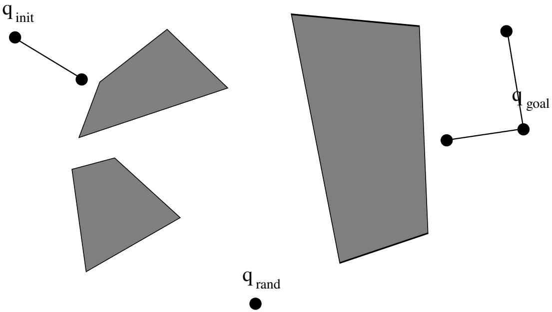
Rapidly exploring Random Tree (RRT) 2000



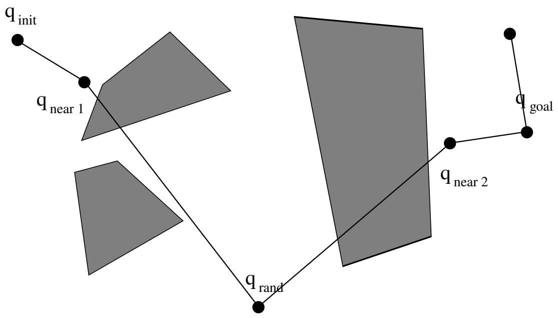
Rapidly exploring Random Tree (RRT) 2000



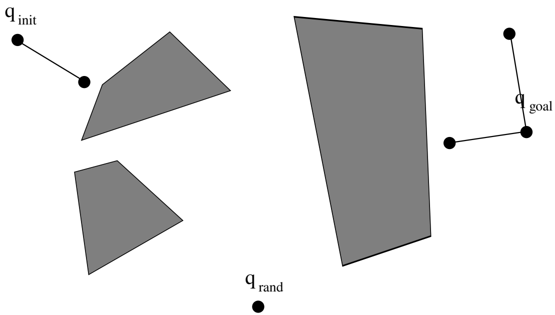
Rapidly exploring Random Tree (RRT) 2000



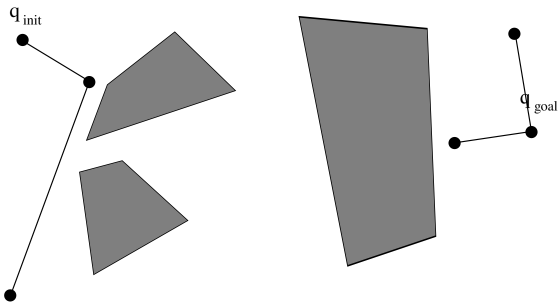
Rapidly exploring Random Tree (RRT) 2000



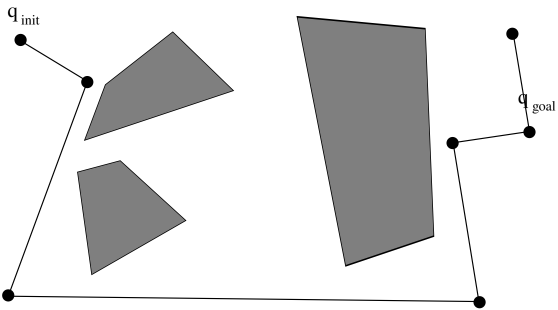
Rapidly exploring Random Tree (RRT) 2000



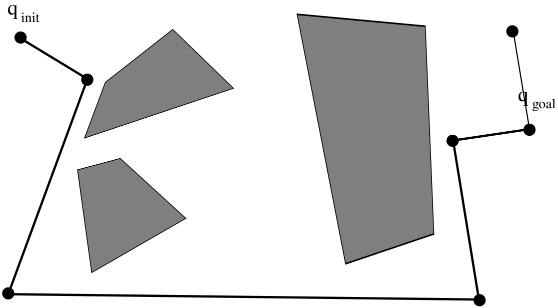
Rapidly exploring Random Tree (RRT) 2000



Rapidly exploring Random Tree (RRT) 2000



Rapidly exploring Random Tree (RRT) 2000



Random methods

- Pros :
  - no explicit computation of the free configuration space,
  - easy to implement,
  - robust.
- Cons :
  - no completeness property, only probabilistic completeness,
  - difficult to find narrow passages.
- Requested operators :
  - Collision tests
    - for configurations (static),
    - for paths (dynamic)

Random methods

- Pros :
  - no explicit computation of the free configuration space,
  - easy to implement,
  - robust.
- Cons :
  - no completeness property, only probabilistic completeness,
  - difficult to find narrow passages.
- Requested operators :
  - Collision tests
    - for configurations (static),
    - for paths (dynamic)

Random methods

- Pros :
  - no explicit computation of the free configuration space,
  - easy to implement,
  - robust.
- Cons :
  - no completeness property, only probabilistic completeness,
  - difficult to find narrow passages.
- Requested operators :
  - Collision tests
    - for configurations (static),
    - for paths (dynamic)

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.

Numerical constraints

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

- Numerical constraints :

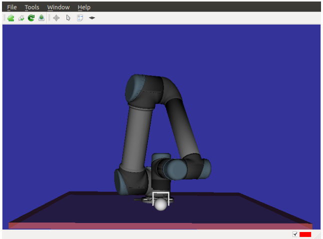
$$f(\mathbf{q}) = 0, \quad \begin{array}{l} m \in \mathbb{N}, \\ f \in C^1(\mathcal{C}, \mathbb{R}^m) \end{array}$$

  - `setConstantRightHandSide(True)`
- Parameterizable numerical constraints :

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

  - `setConstantRightHandSide(False)`

Example : robot manipulating a ball



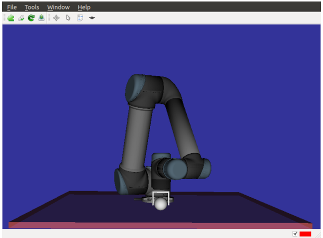
$$\mathcal{C} = [-\pi, \pi]^6 \times \mathbb{R}^3 \quad (1)$$

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

Two *states* :

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

Example : robot manipulating a ball



Each state is defined by a numerical constraint

- placement

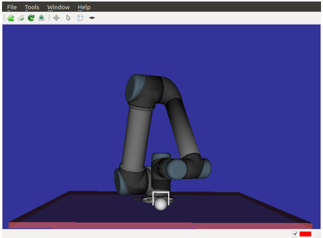
$$z_b = 0$$

- 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`

$$z_b = 0$$

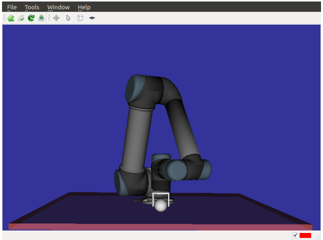
- `grasp`

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

Each state is a sub-manifold of the configuration space

Example : robot manipulating a ball

Motion constraints

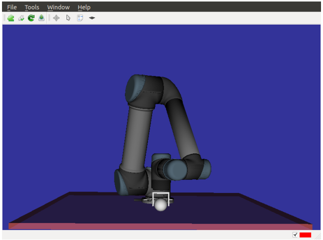


Two *types of motion* :

- `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`

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

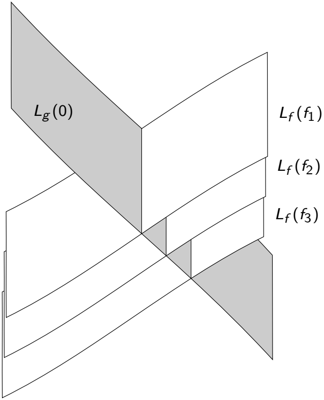
$$\left. z_b = 0 \right\} \text{ simple}$$

- `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

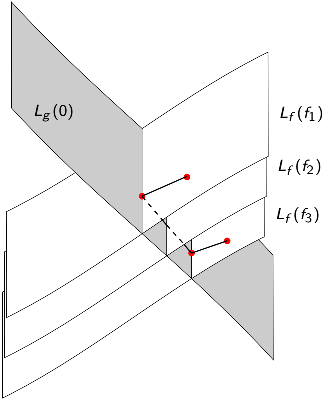
$$L_f(f_1) = \{\mathbf{q} \in \mathcal{C}, f(\mathbf{q}) = f_1\}$$

- 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` $\cup$ `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
- trajectories of the system are subject to
  - numerical constraints
  - parameterizable numerical constraints.

General case

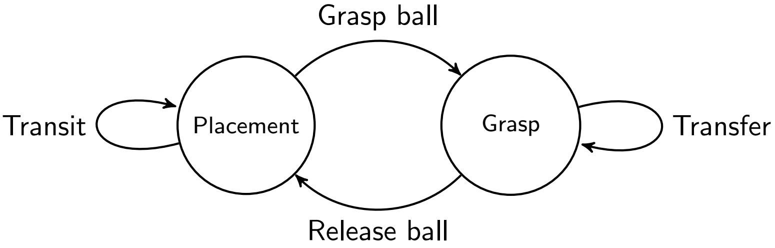
In a manipulation problem,

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

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

Newton-Raphson algorithm

- $\mathbf{q}_0$ 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 :

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

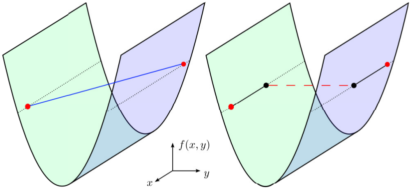
if $\|f(\mathbf{q})\| < \epsilon$: return $\mathbf{q}$

return failure

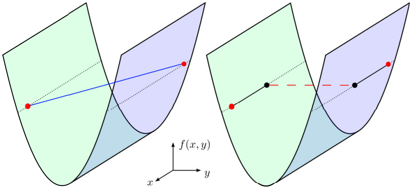
Projecting path on constraint

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

Applying Newton Raphson at each point may result in a discontinuous path



Discontinuous Projection



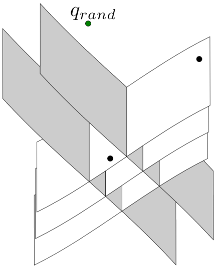
$$\mathcal{C} = \mathbb{R}^2, f(x,y) = y^2 - 1$$

$$\frac{\partial f}{\partial \mathbf{q}} = \begin{pmatrix} 0 & 2y \end{pmatrix}, \ \frac{\partial f}{\partial \mathbf{q}}^+ = \begin{pmatrix} 0 \\ \frac{1}{2y} \end{pmatrix} \text{ ou } \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$y_{i+1} = y_i + \frac{1 - y_i^2}{2y_i}$$

Algorithm

Manipulation RRT



Manipulation RRT

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

for each connected component :

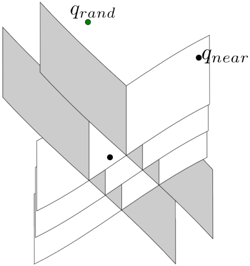
$\mathbf{q}_{near}$ = nearest_neighbor($\mathbf{q}_{rand}$, *roadmap*)
T = select_transition($\mathbf{q}_{near}$)
$\mathbf{q}_{proj}$ = generate_target_config($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$, T)
$\mathbf{q}_{new}$ = extend($\mathbf{q}_{near}$, $\mathbf{q}_{proj}$, T)
roadmap.insert_node($\mathbf{q}_{new}$)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

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

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

Algorithm

Manipulation RRT



Manipulation RRT

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

for each connected component :

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

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

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

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

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

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

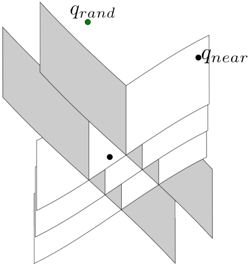
new_nodes.append ($\mathbf{q}_{new}$)

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

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

Algorithm

Manipulation RRT



Manipulation RRT

$\mathbf{q}_{rand} =$ shoot_random_config()
for each connected component :

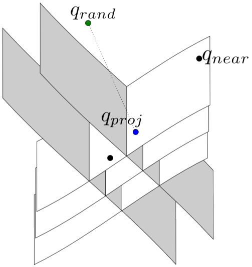
$\mathbf{q}_{near} =$ nearest_neighbor($\mathbf{q}_{rand}$, *roadmap*)
$T =$ select_transition($\mathbf{q}_{near}$)
$\mathbf{q}_{proj} =$ generate_target_config($\mathbf{q}_{near}, \mathbf{q}_{rand}, T$)
$\mathbf{q}_{new} =$ extend($\mathbf{q}_{near}$, $\mathbf{q}_{proj}$, T)
roadmap.insert_node($\mathbf{q}_{new}$)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

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

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

Algorithm

Manipulation RRT



Manipulation RRT

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

for each connected component :

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

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

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

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

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

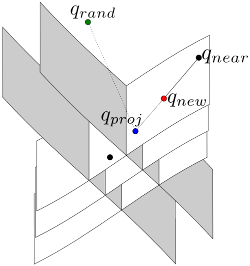
$\quad \textit{roadmap}.\text{insert\_edge}(T, \mathbf{q}_{near}, \mathbf{q}_{new})$

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

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

$\quad$ connect ($\mathbf{q}$, roadmap)

Algorithm
Manipulation RRT



Manipulation RRT

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

for each connected component :

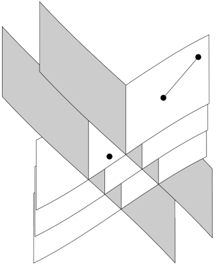
- $\mathbf{q}_{near}$ = nearest_neighbor($\mathbf{q}_{rand}$, *roadmap*)
- T = select_transition($\mathbf{q}_{near}$)
- $\mathbf{q}_{proj}$ = generate_target_config($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$, T)
- $\mathbf{q}_{new}$ = extend($\mathbf{q}_{near}$, $\mathbf{q}_{proj}$, T)
- *roadmap*.insert_node($\mathbf{q}_{new}$)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

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

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

Algorithm

Manipulation RRT



Manipulation RRT

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

for each connected component :

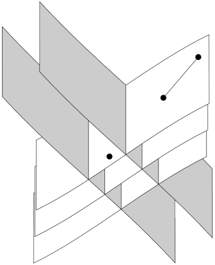
$\mathbf{q}_{near} = \text{nearest\_neighbor}(\mathbf{q}_{rand}, \textit{roadmap})$
$T = \text{select\_transition}(\mathbf{q}_{near})$
$\mathbf{q}_{proj} = \text{generate\_target\_config}(\mathbf{q}_{near}, \mathbf{q}_{rand}, T)$
$\mathbf{q}_{new} = \text{extend}(\mathbf{q}_{near}, \mathbf{q}_{proj}, T)$
$\textit{roadmap}.\text{insert\_node}(\mathbf{q}_{new})$
$\textit{roadmap}.\text{insert\_edge}(T, \mathbf{q}_{near}, \mathbf{q}_{new})$
$\text{new\_nodes.append}(\mathbf{q}_{new})$

for $\mathbf{q} \in (\mathbf{q}^{1}_{new}, \dots, \mathbf{q}^{n_{cc}}_{new})$:

connect (q, roadmap)

Algorithm

Manipulation RRT



Manipulation RRT

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

for each connected component :

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

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

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

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

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

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

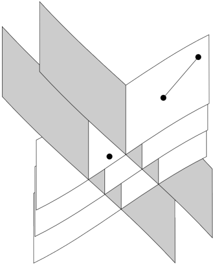
$\quad \text{new\_nodes.append}\ (\mathbf{q}_{new})$

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

$\quad$ connect ($\mathbf{q}$, roadmap)

Algorithm

Manipulation RRT



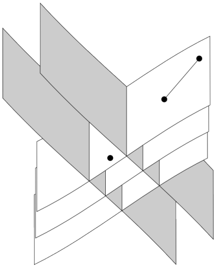
Manipulation RRT

$\mathbf{q}_{rand} =$ shoot_random_config()
for each connected component :

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

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

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



Manipulation RRT

$\mathbf{q}_{rand}$ = shoot_random_config()
for each connected component :

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

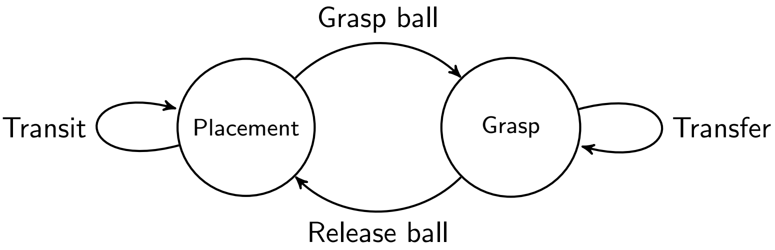
for $\mathbf{q} \in (\mathbf{q}^1_{new}, ..., \mathbf{q}^{n_{cc}}_{new})$:

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

Select transition

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

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



Generate target configuration

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

Once transition T has been selected, $\mathbf{q}_{rand}$ is *projected* onto the destination state S_{dest} in a configuration reachable by $\mathbf{q}_{near}$.

$$f_T(\mathbf{q}_{proj}) = f_T(\mathbf{q}_{near})$$
$$f_{S_{dest}}(\mathbf{q}_{proj}) = 0$$

Extend

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

Project straight path $[\mathbf{q}_{near}, \mathbf{q}_{proj}]$ on transition 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.

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

Extend

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

Project straight path $[\mathbf{q}_{near}, \mathbf{q}_{proj}]$ on transition 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.

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

Extend

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

Project straight path $[\mathbf{q}_{near}, \mathbf{q}_{proj}]$ on transition 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.

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

Connect

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

for each connected component *cc* not containing $\mathbf{q}$:

for all *n* closest config $\mathbf{q}_1$ to $\mathbf{q}$ in *cc* :

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

Connect

```
connect (q_0, q_1) :
    S_0 = state (q_0)
    S_1 = state (q_1)
    T = transition (S_0, S_1)
    if T and f_T(q_0) == f_T(q_1) :
        if p = projected_path (T, q_0, q_1) collision-free :
            roadmap.insert_edge (T, q_0,q_1)
    return
```

Relative positions as numerical constraints

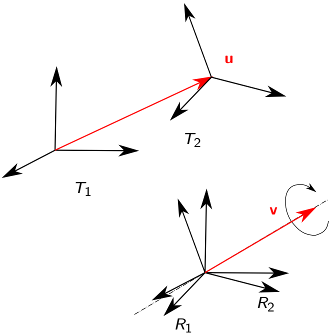
- $T_1 = T_{(R_1,t_1)} \in SE(3)$,
  $T_2 = T_{(R_2,t_2)} \in SE(3)$.
- $T_{2/1} = T_1^{-1} \circ T_2$ can be represented by a vector of dimension 6 :

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

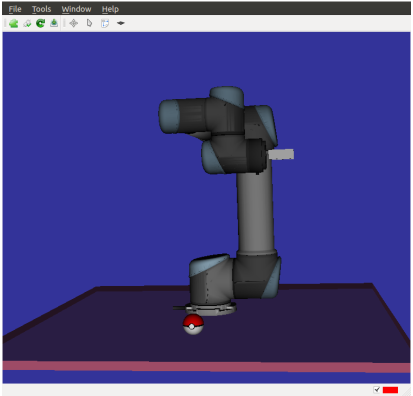
  where

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

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



A few words about the BE



- `script/grasp_ball.py`