Manipulation motion planning

Florent Lamiraux

CNRS-LAAS, Toulouse, France
A few examples

Manipulation motion planning
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.
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

\[ q = (q_0, \ldots, q_{nr}, q_{nr+1}, \ldots, q_{nr+n_0}) \]

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

\[ C = C_{r1} \times C_{nr \text{ robots}} \times SE(3)^{nb \text{ objs}} \]
Composite robot

Kinematic chain composed of each robot and of each object

\[ q = (q_0, \ldots, q_{nr}, q_{nr+1}, \ldots, q_{nr+n_o}) \]

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

\[ C = C_{r1} \times C_{r_{nb \, robots}} \times SE(3)^{nb \, objs} \]
Numerical constraints

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

- Numerical constraints:
  \[ f(q) = 0, \quad m \in \mathbb{N}, \quad f \in C^1(C, \mathbb{R}^m) \]

- Parameterizable numerical constraints:
  \[ f(q) = f_0, \quad m \in \mathbb{N}, \quad f \in C^1(C, \mathbb{R}^m) \]
  \[ f_0 \in \mathbb{R}^m \]
Example: robot manipulating a ball

\[ C = [-\pi, \pi]^6 \times \mathbb{R}^3 \]  
\[ q = (q_0, \ldots, q_5, x_b, y_b, z_b) \]

**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**

\[ x_{gripper}(q_0, \ldots, q_5) - \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = 0 \]

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

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**

\[
\begin{align*}
x_b &= x_0 \\
y_b &= y_0 \\
z_b &= 0
\end{align*}
\] (parameterizable)

- **transfer**

\[
x_{gripper}(q_0, \ldots, q_5) - \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = 0
\]
Motion constraints define a foliation of the admissible configuration space \((\text{grasp} \cup \text{placement})\).

- \(f\): position of the ball
  \[
  L_f(f_1) = \{q \in C, f(q) = f_1\}
  \]

- \(g\): grasp of the ball
  \[
  L_g(0) = \{q \in C, g(q) = 0\}
  \]
Foliation

Motion constraints define a foliation of the admissible configuration space \((\text{grasp } \cup \text{ placement})\).

\[
L_g(0) \quad L_f(f_1) \quad L_f(f_2) \quad L_f(f_3)
\]

Solution to a manipulation planning problem is a concatenation of \textit{transit} and \textit{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.
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, 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
▶ trajectories of the system 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

Newton-Raphson algorithm

- $q_0$ configuration,
- $f(q) = 0$ non-linear constraint,
- $\epsilon$ numerical tolerance

Projection $(q_0, f)$:

$q = q_0; \alpha = 0.95$

for $i$ from 1 to max_iter:

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

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

return failure
Steering method

Mapping $SM$ from $C \times C$ to $C^1([0, 1], C)$ such that

$$SM(q_0, q_1)(0) = q_0$$
$$SM(q_0, q_1)(1) = q_1$$
Constrained steering method

Let

- $SM$ be a steering method
- $f \in C^1(C, \mathbb{R}^m)$ be a numerical constraint.

A constrained steering method $SM$ of constraint $f$ is a steering method such that

$$\forall t \in [0, 1], f(SM(t)) = 0$$
Projecting path on constraint

- path: mapping from $[0, 1]$ to $C$
- $f(q) = 0$ non-linear constraint,

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

\[ C = \mathbb{R}^2, \quad f(x, y) = y^2 - 1 \]

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

\[ y_{i+1} = y_i + \frac{1 - y_i^2}{2y_i} \]
Testing projection continuity

- The initial path is sampled and successive samples are projected,
Testing projection continuity

- The initial path is sampled and successive samples are projected,
Testing projection continuity

- The initial path is sampled and successive samples are projected,
Testing projection continuity

- The initial path is sampled and successive samples are projected,
- if 2 successive projections are too far away, an intermediate sample is selected.
Testing projection continuity

- The initial path is sampled and successive samples are projected,
- if 2 successive projections are too far away, an intermediate sample is selected.
- Choosing appropriate sampling ensures us continuity of the projection.
Algorithm
Manipulation RRT

Manipulation RRT

\( q_{\text{rand}} = \text{shoot\_random\_config}() \)

for each connected component:

\( q_{\text{near}} = \text{nearest\_neighbour}(q_{\text{rand}}, \text{roadmap}) \)

\( e = \text{select\_transition}(q_{\text{near}}) \)

\( q_{\text{proj}} = \text{generate\_target\_config}(q_{\text{near}}, q_{\text{rand}}, e) \)

\( q_{\text{new}} = \text{extend}(q_{\text{near}}, q_{\text{proj}}, e) \)

\( \text{roadmap}.\text{insert\_node}(q_{\text{new}}) \)

\( \text{roadmap}.\text{insert\_edge}(e, q_{\text{near}}, q_{\text{new}}) \)

new\_nodes.append (q_{\text{new}})

for \( q \in (q_{\text{new}}^{1}, \ldots, q_{\text{new}}^{n_{\text{cc}}}) \):

connect (q, roadmap)

Manipulation motion planning
Algorithm
Manipulation RRT

**Manipulation RRT**

\( q_{rand} = \text{shoot\_random\_config}() \)

for each connected component:

\( q_{near} = \text{nearest\_neighbour}(q_{rand}, \text{roadmap}) \)

\( e = \text{select\_transition}(q_{near}) \)

\( q_{proj} = \text{generate\_target\_config}(q_{near}, q_{rand}, e) \)

\( q_{new} = \text{extend}(q_{near}, q_{proj}, e) \)

\( \text{roadmap.insert\_node}(q_{new}) \)

\( \text{roadmap.insert\_edge}(e, q_{near}, q_{new}) \)

\( \text{new\_nodes.append}(q_{new}) \)

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

\( \text{connect}(q, \text{roadmap}) \)
Algorithm
Manipulation RRT

Manipulation RRT

$q_{rand} = \text{shoot\_random\_config}()$

for each connected component:

$q_{near} = \text{nearest\_neighbour}(q_{rand}, \text{roadmap})$

e = \text{select\_transition}(q_{near})

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

$q_{new} = \text{extend}(q_{near}, q_{proj}, e)$

$\text{roadmap.insert\_node}(q_{new})$

$\text{roadmap.insert\_edge}(e, q_{near}, q_{new})$

new\_nodes.append (q_{new})

for $q \in (q_{new}^1, \ldots, q_{new}^{ncc})$:

connect ($q$, roadmap)
Algorithm
Manipulation RRT

Manipulation RRT

$q_{rand} = \text{shoot\_random\_config}()$

for each connected component:

$q_{near} = \text{nearest\_neighbour}(q_{rand}, \text{roadmap})$

e = \text{select\_transition}(q_{near})

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

$q_{new} = \text{extend}(q_{near}, q_{proj}, e)$

roadmap.insert_node($q_{new}$)

roadmap.insert_edge(e, $q_{near}, q_{new}$)

new\_nodes.append($q_{new}$)

for $q \in (q_{new}^{1}, ..., q_{new}^{ncc})$:

connect($q$, roadmap)
Algorithm
Manipulation RRT

Manipulation RRT

\( q_{rand} = \text{shoot\_random\_config()} \)

for each connected component:

\( q_{near} = \text{nearest\_neighbour}(q_{rand}, \text{roadmap}) \)

\( e = \text{select\_transition}(q_{near}) \)

\( q_{proj} = \text{generate\_target\_config}(q_{near}, q_{rand}, e) \)

\( q_{new} = \text{extend}(q_{near}, q_{proj}, e) \)

roadmap.insert_node(\( q_{new} \))

roadmap.insert_edge(\( e, q_{near}, q_{new} \))

new_nodes.append(\( q_{new} \))

for \( q \in (q_{new}^1, \ldots, q_{new}^{ncc}) \):

connect(\( q, \text{roadmap} \))
Algorithm
Manipulation RRT

Manipulation RRT

\( q_{\text{rand}} = \text{shoot\_random\_config()} \)

for each connected component:

\( q_{\text{near}} = \text{nearest\_neighbour}(q_{\text{rand}}, \text{roadmap}) \)
\( e = \text{select\_transition}(q_{\text{near}}) \)
\( q_{\text{proj}} = \text{generate\_target\_config}(q_{\text{near}}, q_{\text{rand}}, e) \)
\( q_{\text{new}} = \text{extend}(q_{\text{near}}, q_{\text{proj}}, e) \)

\text{roadmap}.insert\_node(q_{\text{new}}) \\
\text{roadmap}.insert\_edge(e, q_{\text{near}}, q_{\text{new}}) \\
\text{new\_nodes}.append(q_{\text{new}}) \\

for \( q \in (q_{\text{new}}^{1}, \ldots, q_{\text{new}}^{n_{\text{cc}}}) \):

\text{connect}(q, \text{roadmap})
Algorithm
Manipulation RRT

Manipulation RRT

\( q_{\text{rand}} = \text{shoot\_random\_config()} \)

for each connected component:

\[ q_{\text{near}} = \text{nearest\_neighbour}(q_{\text{rand}}, \text{roadmap}) \]

\[ e = \text{select\_transition}(q_{\text{near}}) \]

\[ q_{\text{proj}} = \text{generate\_target\_config}(q_{\text{near}}, q_{\text{rand}}, e) \]

\[ q_{\text{new}} = \text{extend}(q_{\text{near}}, q_{\text{proj}}, e) \]

\[ \text{roadmap}.\text{insert\_node}(q_{\text{new}}) \]

\[ \text{roadmap}.\text{insert\_edge}(e, q_{\text{near}}, q_{\text{new}}) \]

\[ \text{new\_nodes}.\text{append}(q_{\text{new}}) \]

for \( q \in (q_{\text{new}}^1, \ldots, q_{\text{new}}^{n_{\text{cc}}}) \):

\[ \text{connect}(q, \text{roadmap}) \]
Algorithm
Manipulation RRT

Manipulation RRT

$q_{rand} = \text{shoot\_random\_config}()$

for each connected component:

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

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

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

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

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

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

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

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

connect(\( q \), roadmap)
Algorithm
Manipulation RRT

Manipulation RRT

$q_{\text{rand}} = \text{shoot\_random\_config}()$

for each connected component:

$q_{\text{near}} = \text{nearest\_neighbour}(q_{\text{rand}}, \text{roadmap})$

$e = \text{select\_transition}(q_{\text{near}})$

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

$q_{\text{new}} = \text{extend}(q_{\text{near}}, q_{\text{proj}}, e)$

$\text{roadmap}.\text{insert\_node}(q_{\text{new}})$

$\text{roadmap}.\text{insert\_edge}(e, q_{\text{near}}, q_{\text{new}})$

$\text{new\_nodes}.\text{append}(q_{\text{new}})$

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

$\text{connect}(q, \text{roadmap})$
Select transition

e = select_transition(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

\[ q_{proj} = \text{generate\_target\_config}(q_{near}, q_{rand}, e) \]

Once edge \( e \) has been selected, \( q_{rand} \) is projected onto the destination node \( n_{dest} \) in a configuration reachable by \( q_{near} \).

\[
\begin{align*}
    f_e(q_{proj}) &= f_e(q_{near}) \\
    f_{dest}(q_{proj}) &= 0
\end{align*}
\]
Extend

$q_{new} = \text{extend}(q_{near}, q_{proj}, \text{edge})$

*Project* straight path $[q_{near}, q_{proj}]$ on edge constraint:

- if projection successful and projected path collision free

  \[q_{new} \leftarrow q_{proj}\]

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

\[\forall q \in (q_{near}, q_{new}), \quad f_e(q) = f_e(q_{near})\]
\[ q_{\text{new}} = \text{extend}(q_{\text{near}}, q_{\text{proj}}, \text{edge}) \]

*Project* straight path \([q_{\text{near}}, q_{\text{proj}}]\) on edge constraint:

- if projection successful and projected path collision free

  \[ q_{\text{new}} \leftarrow q_{\text{proj}} \]

- otherwise \((q_{\text{near}}, q_{\text{new}}) \leftarrow \text{largest path interval tested as collision-free with successful projection.}\)

\[ \forall q \in (q_{\text{near}}, q_{\text{new}}), \ f_e(q) = f_e(q_{\text{near}}) \]
Extend

$q_{new} = \text{extend}(q_{near}, q_{proj}, \text{edge})$

*Project* straight path $[q_{near}, q_{proj}]$ on edge constraint:

- if projection successful and projected path collision free

\[ q_{new} \leftarrow q_{proj} \]

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

\[
\forall q \in (q_{near}, q_{new}),
\quad f_e(q) = f_e(q_{near})
\]
Connect

connect \((q, \text{roadmap})\)

for each connected component \(cc\) not containing \(q\):
for all \(n\) closest config \(q_1\) to \(q\) in \(cc\):
  ▶ connect \((q, q_1)\)
connect \((q_0, q_1)\):

\[ s_0 = \text{state} \ (q_0) \]
\[ s_1 = \text{state} \ (q_1) \]
\[ e = \text{transition} \ (n_0, n_1) \]

if \(e\) and \(f_e(q_0) == f_e(q_1)\):

if \(p = \text{projected\_path} \ (e, q_0, q_1)\) collision-free:

roadmap.insert_edge \((e, q_0, q_1)\)

return
Connecting trees

Manipulation RRT is initialized with $q_{init}$, $q_{goal}$.

- 2 connected components.
- Possible connection.
Connecting trees: general case

Manipulation RRT is initialized with $q_{\text{init}}$, $q_{\text{goal}}$.

- 2 connected components,
- no possible connection.
Connecting trees: general case

Manipulation RRT is initialized with \( q_{init}, q_{goal} \).

- 2 connected components,
- no possible connection.

Manipulation motion planning
Crossed foliation transition: generate target configuration

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

\[ \mathbf{q}_1 \leftarrow \text{pick configuration} \]

- in node \( N_1 \),
- not in same connected component as \( \mathbf{q}_{near} \)

\[ f_{e_1}(\mathbf{q}_{proj}) = f_{e_1}(\mathbf{q}_{near}) \]
\[ f_{e_2}(\mathbf{q}_{proj}) = f_{e_2}(\mathbf{q}_1) \]
\[ f_{N_2}(\mathbf{q}_{proj}) = 0 \]
Crossed foliation transition: extend

\[ q_{new} = \text{extend}(q_{near}, q_{proj}, e_1) \]

*Project* straight path \([q_{near}, q_{proj}]\) on \(e_1\) constraint:

- if projection successful and projected path collision free

\[ q_2 \leftarrow q_{proj} \]

\[ f_{e_2}(q_2) = f_{e_2}(q_1) \]
\[ f_{N_2}(q_2) = 0 \]

- \(q_2\) is connectable to \(q_1\) via \(e_2\).
Crossed foliation transition: extend

\[ q_{new} = \text{extend}(q_{\text{near}}, q_{\text{proj}}, e_1) \]

*Project* straight path \([q_{\text{near}}, q_{\text{proj}}]\) on \(e_1\) constraint:

- if projection successful and projected path collision free

\[ q_2 \leftarrow q_{\text{proj}} \]

\[ f_{e_2}(q_2) = f_{e_2}(q_1) \]
\[ f_{N_2}(q_2) = 0 \]

- \(q_2\) is connectable to \(q_1\) via \(e_2\).
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:

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

where

$$u = R_1^T (t_2 - t_1)$$

$R_1^T R_2$ is the matrix of the rotation around axis $v/\|v\|$ and of angles $\|v\|$. 
A few words about the BE

Manipulation motion planning