Introduction to Robotics

Manipulation Planning

Erion Plaku

Department of Electrical Engineering and Computer Science
Catholic University of America

Problem Formulation

Given [movie: L-shape] [movie: industrial]

- a description of the obstacles
- a description of the robot manipulator
- a description of the object to be manipulated
- a description of the initial and desired placements for the object

compute a sequence of motions where the robot manipulator grasps the object in its initial placement and places it in its desired placement while avoding collisions

Problem Formulation

Given [movie: L-shape] [movie: industrial]

- a description of the obstacles
- a description of the robot manipulator
- a description of the object to be manipulated
- a description of the initial and desired placements for the object

compute a sequence of motions where the robot manipulator grasps the object in its initial placement and places it in its desired placement while avoding collisions

Challenges

- How to grasp the object? Is the grasp stable?
- Does the solution require re-grasping? When should the robot manipulator release the object and re-grasp it in a different configuration?

Observations

- Solution path to manipulation-planning problem consists of a sequence of transfer and transit paths
- Transfer path is a subpath where the object is stably grasped and moved by the robot
- Transit path is a subpath where the object is left in a stable position while the robot changes grasp

Each node is a triple $(q_{\mathrm{obj}}, g, q_{\mathrm{rob}})$, where

- q_{obj} specifies a stable placement (position and orientation) of the object
- g specifies a position and orientation of the robot tool relative to the placement of the object at which the tool is able to grasp the object
- q_{rob} is the configuration of the robot for which the robot tool is able to grasp the object placed at q_{obj} using the grasp g

Each node is a triple $(q_{\mathrm{obj}}, g, q_{\mathrm{rob}})$, where

- q_{obj} specifies a stable placement (position and orientation) of the object
- g specifies a position and orientation of the robot tool relative to the placement of the object at which the tool is able to grasp the object
- q_{rob} is the configuration of the robot for which the robot tool is able to grasp the object placed at q_{obj} using the grasp g

An edge $\left((q^i_{\mathrm{obj}}, g, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g, q^j_{\mathrm{rob}})\right)$ indicates a tranfer path where the object is grasped according to g and the robot moves with the object from configuration $(q^i_{\mathrm{obj}}, q^i_{\mathrm{rob}})$ to $(q^j_{\mathrm{obj}}, q^j_{\mathrm{rob}})$

Each node is a triple $(q_{\mathrm{obj}}, g, q_{\mathrm{rob}})$, where

- q_{obj} specifies a stable placement (position and orientation) of the object
- g specifies a position and orientation of the robot tool relative to the placement of the object at which the tool is able to grasp the object
- q_{rob} is the configuration of the robot for which the robot tool is able to grasp the object placed at q_{obj} using the grasp g

An edge $\left((q^i_{\mathrm{obj}}, g, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g, q^j_{\mathrm{rob}})\right)$ indicates a tranfer path where the object is grasped according to g and the robot moves with the object from configuration $(q^i_{\mathrm{obj}}, q^i_{\mathrm{rob}})$ to $(q^j_{\mathrm{obj}}, q^j_{\mathrm{rob}})$

An edge $\left((q_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})\right)$ indicates a transit path where the object is left at a stable placement q_{obj} while the robot changes grasp from $(g^i, q^i_{\mathrm{rob}})$ to $(g_j, q^j_{\mathrm{rob}})$

Computing the Manipulation Graph

PRM Approach

- Node Generation:

  for $i = 1, \ldots, N$ **do** sample a node $(q_{\mathrm{obj}}^{i}, g^{i}, q_{\mathrm{rob}}^{i})$

PRM Approach

- Node Generation:

  for $i = 1, \ldots, N$ **do** sample a node $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$

- Edge Generation:

  connect neighboring nodes $\left((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})\right)$

PRM Approach

- Node Generation:

$$\textbf{for } i = 1, \ldots, N \textbf{ do} \text{ sample a node } (q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$$

- Edge Generation:

$$\text{connect neighboring nodes } \left((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})\right)$$

Challenges with PRM Approach

- Each edge generation gives rise to a path-planning problem
- Must verify edge validity before adding it to manipulation graph
- Too many edge verifications (since graph could have large number of nodes)

PRM Approach

- Node Generation:

  for $i = 1, \ldots, N$ **do** sample a node $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$

- Edge Generation:

  connect neighboring nodes $\left((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})\right)$

Challenges with PRM Approach

- Each edge generation gives rise to a path-planning problem
- Must verify edge validity before adding it to manipulation graph
- Too many edge verifications (since graph could have large number of nodes)

FuzzyPRM Idea

- Probabilistic edges instead of deterministic edges
- Use a probabilistic path planner to compute edge connections
- Probability associated with an edge e depends on the time spent by probabilistic path planner on e

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
2: **for** each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$,
$i = 1, \ldots, N$ of the manipulation graph
2: **for** each pair of nodes
$e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
3: **if** $g^i = g^j$ **then** add e as a transfer edge
and set $prob(e) \leftarrow 0.9999$

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
2: **for** each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
3: **if** $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
4: **if** $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
2: **for** each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
3: **if** $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
4: **if** $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

1: **while** no solution found **do**

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
2: **for** each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
3: **if** $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
4: **if** $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

1: **while** no solution found **do**
2: $\sigma \leftarrow$ compute most probable path in the manipulation graph

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
2: **for** each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
3: **if** $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
4: **if** $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

1: **while** no solution found **do**
2: $\sigma \leftarrow$ compute most probable path in the manipulation graph
3: **for** each edge $e \in \sigma$ **do**

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

```
User supplies nodes (q^i_obj, g^i, q^i_rob),
i = 1, ..., N of the manipulation graph
for each pair of nodes
e = ((q^i_obj, g^i, q^i_rob), (q^j_obj, g^j, q^j_rob)) do
    if g^i = g^j then add e as a transfer edge
    and set prob(e) ← 0.9999
    if q^i_obj = q^j_obj then add e as a transit
    edge and set prob(e) ← 0.9999
```

- User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
- **for** each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}}))$ **do**
  - **if** $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
  - **if** $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

- **while** no solution found **do**
  - $\sigma \leftarrow$ compute most probable path in the manipulation graph
  - **for** each edge $e \in \sigma$ **do**
    - **if** $prob(e) \neq 1$ **then**
      - run low-level fuzzy PRM on e for a short period of time

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
for each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
    if $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
    if $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

while no solution found **do**
    $\sigma \leftarrow$ compute most probable path in the manipulation graph
    for each edge $e \in \sigma$ **do**
        if $prob(e) \neq 1$ **then**
            run low-level fuzzy PRM on e for a short period of time
            if success **then**
                $prob(e) \leftarrow 1$

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
for each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
    if $g^i = g^j$ **then** add e as a transfer edge and set $\mathit{prob}(e) \leftarrow 0.9999$
    if $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $\mathit{prob}(e) \leftarrow 0.9999$

Query Stage

while no solution found **do**
    $\sigma \leftarrow$ compute most probable path in the manipulation graph
    for each edge $e \in \sigma$ **do**
        if $\mathit{prob}(e) \neq 1$ **then**
            run low-level fuzzy PRM on e for a short period of time
            if success **then**
                $\mathit{prob}(e) \leftarrow 1$
            else
                $\mathit{prob}(e) \leftarrow \frac{1 - \mathit{time}(e)}{\mathit{total\_time}}$

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
for each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
  if $g^i = g^j$ **then** add e as a transfer edge and set $\mathit{prob}(e) \leftarrow 0.9999$
  if $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $\mathit{prob}(e) \leftarrow 0.9999$

Query Stage

while no solution found **do**
  $\sigma \leftarrow$ compute most probable path in the manipulation graph
  for each edge $e \in \sigma$ **do**
    if $\mathit{prob}(e) \neq 1$ **then**
      run low-level fuzzy PRM on e for a short period of time
      if success **then**
        $\mathit{prob}(e) \leftarrow 1$
      else
        $\mathit{prob}(e) \leftarrow \frac{1 - \mathit{time}(e)}{\mathit{total\_time}}$

Low-Level Fuzzy PRM

if mode = "CONSTRUCTION" **then**
  add a new sample q to graph G_e
  add an edge(q, q') to all previous samples
  $\mathit{prob}(q, q') \leftarrow P^*(l)$

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
2: **for** each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
3:    **if** $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
4:    **if** $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

1: **while** no solution found **do**
2:    $\sigma \leftarrow$ compute most probable path in the manipulation graph
3:    **for** each edge $e \in \sigma$ **do**
4:       **if** $prob(e) \neq 1$ **then**
5:          run low-level fuzzy PRM on e for a short period of time
6:          **if** success **then**
7:             $prob(e) \leftarrow 1$
8:          **else**
9:             $prob(e) \leftarrow \frac{1 - time(e)}{total\_time}$

Low-Level Fuzzy PRM

1: **if** mode = "CONSTRUCTION" **then**
2:    add a new sample q to graph G_e
3:    add an edge(q, q') to all previous samples
4:    $prob(q, q') \leftarrow P^*(l)$
5: **if** mode = "QUERY" **then**

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
for each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
    if $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
    if $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

while no solution found **do**
    $\sigma \leftarrow$ compute most probable path in the manipulation graph
    for each edge $e \in \sigma$ **do**
        if $prob(e) \neq 1$ **then**
            run low-level fuzzy PRM on e for a short period of time
            if success **then**
                $prob(e) \leftarrow 1$
            else
                $prob(e) \leftarrow \frac{1 - time(e)}{total\_time}$

Low-Level Fuzzy PRM

if mode = "CONSTRUCTION" **then**
    add a new sample q to graph G_e
    add an edge(q, q') to all previous samples
    $prob(q, q') \leftarrow P^*(l)$
if mode = "QUERY" **then**
    $\phi \leftarrow$ compute most probable path in G_e

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
2: **for** each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
3:     **if** $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
4:     **if** $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

1: **while** no solution found **do**
2:     $\sigma \leftarrow$ compute most probable path in the manipulation graph
3:     **for** each edge $e \in \sigma$ **do**
4:         **if** $prob(e) \neq 1$ **then**
5:             run low-level fuzzy PRM on e for a short period of time
6:             **if** success **then**
7:                 $prob(e) \leftarrow 1$
8:             **else**
9:                 $prob(e) \leftarrow \frac{1 - time(e)}{total\_time}$

Low-Level Fuzzy PRM

1: **if** mode = "CONSTRUCTION" **then**
2:     add a new sample q to graph G_e
3:     add an edge(q, q') to all previous samples
4:     $prob(q, q') \leftarrow P^*(l)$
5: **if** mode = "QUERY" **then**
6:     $\phi \leftarrow$ compute most probable path in G_e
7:     **repeat**
8:         $(q', q'') \leftarrow$ edge in ϕ with lowest probability

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
for each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
    if $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
    if $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

while no solution found **do**
    $\sigma \leftarrow$ compute most probable path in the manipulation graph
    for each edge $e \in \sigma$ **do**
        if $prob(e) \neq 1$ **then**
            run low-level fuzzy PRM on e for a short period of time
            if success **then**
                $prob(e) \leftarrow 1$
            else
                $prob(e) \leftarrow \frac{1 - time(e)}{total\_time}$

Low-Level Fuzzy PRM

if mode = "CONSTRUCTION" **then**
    add a new sample q to graph G_e
    add an edge(q, q') to all previous samples
    $prob(q, q') \leftarrow P^*(l)$
if mode = "QUERY" **then**
    $\phi \leftarrow$ compute most probable path in G_e
    repeat
        $(q', q'') \leftarrow$ edge in ϕ with lowest probability
        if $prob(q', q'') \neq 1$ **then**
            run subdivision collision checking to validate (q', q'') at resolution $\ell(q', q'')$
            increment $\ell(q', q'')$

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
for each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
  if $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
  if $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

while no solution found **do**
  $\sigma \leftarrow$ compute most probable path in the manipulation graph
  for each edge $e \in \sigma$ **do**
    if $prob(e) \neq 1$ **then**
      run low-level fuzzy PRM on e for a short period of time
      if success **then**
        $prob(e) \leftarrow 1$
      else
        $prob(e) \leftarrow \frac{1 - time(e)}{total\_time}$

Low-Level Fuzzy PRM

if mode = "CONSTRUCTION" **then**
  add a new sample q to graph G_e
  add an edge(q, q') to all previous samples
  $prob(q, q') \leftarrow P^*(l)$
if mode = "QUERY" **then**
  $\phi \leftarrow$ compute most probable path in G_e
  repeat
    $(q', q'') \leftarrow$ edge in ϕ with lowest probability
    if $prob(q', q'') \neq 1$ **then**
      run subdivision collision checking to validate (q', q'') at resolution $\ell(q', q'')$
      increment $\ell(q', q'')$
      if collision **then**
        remove (q', q'') from G_e and return `failure`

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

1: User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
2: **for** each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
3:     **if** $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
4:     **if** $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $prob(e) \leftarrow 0.9999$

Query Stage

1: **while** no solution found **do**
2:     $\sigma \leftarrow$ compute most probable path in the manipulation graph
3:     **for** each edge $e \in \sigma$ **do**
4:         **if** $prob(e) \neq 1$ **then**
5:             run low-level fuzzy PRM on e for a short period of time
6:             **if** success **then**
7:                 $prob(e) \leftarrow 1$
8:             **else**
9:                 $prob(e) \leftarrow \frac{1 - time(e)}{total\_time}$

Low-Level Fuzzy PRM

1: **if** mode = "CONSTRUCTION" **then**
2:     add a new sample q to graph G_e
3:     add an edge(q, q') to all previous samples
4:     $prob(q, q') \leftarrow P^*(l)$
5: **if** mode = "QUERY" **then**
6:     $\phi \leftarrow$ compute most probable path in G_e
7:     **repeat**
8:         $(q', q'') \leftarrow$ edge in ϕ with lowest probability
9:         **if** $prob(q', q'') \neq 1$ **then**
10:             run subdivision collision checking to validate (q', q'') at resolution $\ell(q', q'')$
11:             increment $\ell(q', q'')$
12:             **if** collision **then**
13:                 remove (q', q'') from G_e and return `failure`
14:             **else**
15:                 update $prob(q', q'')$ based on collision resolution $\ell(q', q'')$

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

User supplies nodes $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
for each pair of nodes $e = ((q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g^j, q^j_{\mathrm{rob}})$ **do**
  if $g^i = g^j$ **then** add e as a transfer edge and set $\mathit{prob}(e) \leftarrow 0.9999$
  if $q^i_{\mathrm{obj}} = q^j_{\mathrm{obj}}$ **then** add e as a transit edge and set $\mathit{prob}(e) \leftarrow 0.9999$

Query Stage

while no solution found **do**
  $\sigma \leftarrow$ compute most probable path in the manipulation graph
  for each edge $e \in \sigma$ **do**
    if $\mathit{prob}(e) \neq 1$ **then**
      run low-level fuzzy PRM on e for a short period of time
      if success **then**
        $\mathit{prob}(e) \leftarrow 1$
      else
        $\mathit{prob}(e) \leftarrow \frac{1 - \mathit{time}(e)}{\mathit{total\_time}}$

Low-Level Fuzzy PRM

if mode = "CONSTRUCTION" **then**
  add a new sample q to graph G_e
  add an edge(q, q') to all previous samples
  $\mathit{prob}(q, q') \leftarrow P^*(l)$
if mode = "QUERY" **then**
  $\phi \leftarrow$ compute most probable path in G_e
  repeat
    $(q', q'') \leftarrow$ edge in ϕ with lowest probability
    if $\mathit{prob}(q', q'') \neq 1$ **then**
      run subdivision collision checking to validate (q', q'') at resolution $\ell(q', q'')$
      increment $\ell(q', q'')$
      if collision **then**
        remove (q', q'') from G_e and return `failure`
      else
        update $\mathit{prob}(q', q'')$ based on collision resolution $\ell(q', q'')$
  until all edges in ϕ have prob 1
  return `success`

[Berenson, Srinivasa, Ferguson, Collet, Kuffner: ICRA 2009]

- Manipulation planners often require specification of a set of stable grasp configurations

Manipulation Planning with Workspace Goal Regions

[Berenson, Srinivasa, Ferguson, Collet, Kuffner: ICRA 2009]

- Manipulation planners often require specification of a set of stable grasp configurations
- This forces the planner to use only these configurations as goals
- If the chosen goal configurations are unreachable, the planner will fail
- Even when some of these goal configurations are reachable, it may take the planner a long time to find solutions to these goal configurations

Manipulation Planning with Workspace Goal Regions

[Berenson, Srinivasa, Ferguson, Collet, Kuffner: ICRA 2009]

- Manipulation planners often require specification of a set of stable grasp configurations
- This forces the planner to use only these configurations as goals
- If the chosen goal configurations are unreachable, the planner will fail
- Even when some of these goal configurations are reachable, it may take the planner a long time to find solutions to these goal configurations

Proposed Approach

- Introduce concept of Workspace Goal Regions (WGRs)
- WGR allows the specification of continuous regions in the six-dimensional workspace of end-effector poses as goals for the planner

[Berenson, Srinivasa, Ferguson, Collet, Kuffner: ICRA 2009]

- Manipulation planners often require specification of a set of stable grasp configurations
- This forces the planner to use only these configurations as goals
- If the chosen goal configurations are unreachable, the planner will fail
- Even when some of these goal configurations are reachable, it may take the planner a long time to find solutions to these goal configurations

Proposed Approach

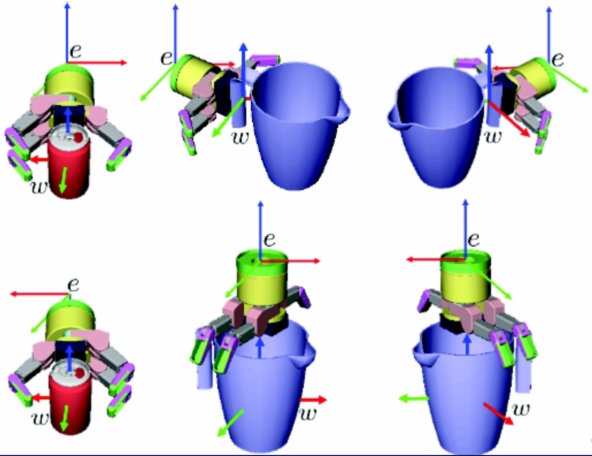
- Introduce concept of Workspace Goal Regions (WGRs)
- WGR allows the specification of continuous regions in the six-dimensional workspace of end-effector poses as goals for the planner
- Desired properties of a WGR
  - easy to describe
  - easy to sample
  - easy to define distance from robot configuration to WGR

[movie]

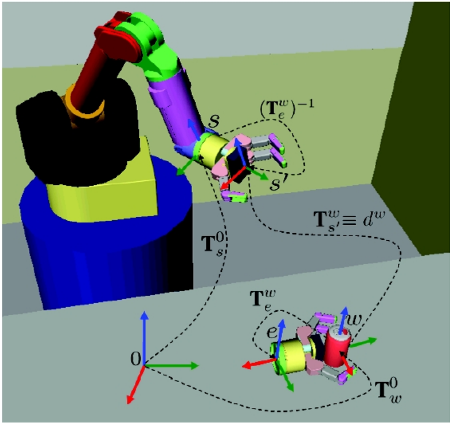
Definition of WGR: a triple (T^0_w, T^e_w, B^w), where

- T^0_w: reference transform of the WGR in world coordinates
- T^e_w: end-effector transform in the coordinates of w
- B^w: bounds in the coordinates of w

$$B^w = [(x_{\min} x_{\max}), (y_{\min}, y_{\max}), (z_{\min}, z_{\max}), (\psi_{\min}, \psi_{\max}), (\theta_{\min}, \theta_{\max}), (\phi_{\min}, \phi_{\max})]$$

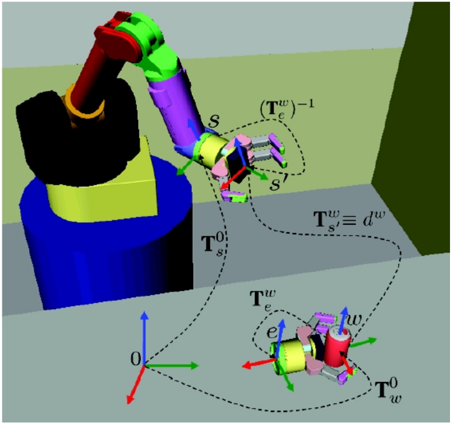


Distance to WGRs: $d(q_s, WGR)$



- use forward kinematics to get the position of the end effector at this configuration T_s^0

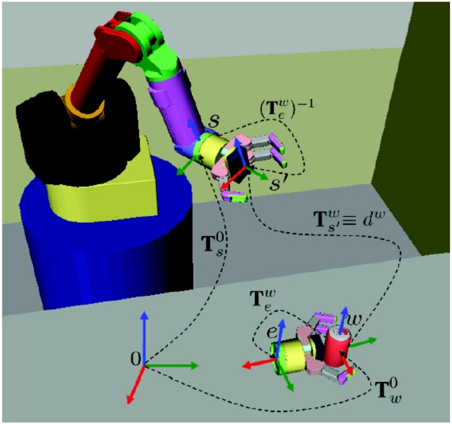
Distance to WGRs: $d(q_s, WGR)$



- use forward kinematics to get the position of the end effector at this configuration T_s^0
- get the pose of the grasp location in world coordinates

$$T_{s'}^0 = T_s^0 (T_e^w)^{-1}$$

Distance to WGRs: $d(q_s, WGR)$



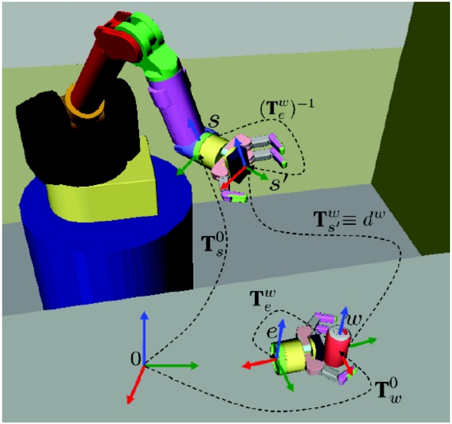
- use forward kinematics to get the position of the end effector at this configuration T_s^0
- get the pose of the grasp location in world coordinates

$$T_{s'}^0 = T_s^0 (T_e^w)^{-1}$$

- convert this pose from world coordinates to the coordinates of w

$$T_{s'}^w = (T_w^0)^{-1} T_{s'}^0$$

Distance to WGRs: $d(q_s, WGR)$



- use forward kinematics to get the position of the end effector at this configuration T^0_s
- get the pose of the grasp location in world coordinates

$$T^0_{s'} = T^0_s (T^w_e)^{-1}$$

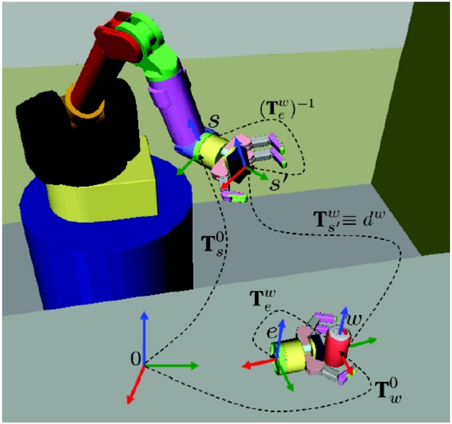
- convert this pose from world coordinates to the coordinates of w

$$T^w_{s'} = (T^0_w)^{-1} T^0_{s'}$$

- convert $T^w_{s'}$ into a 6×1 displacement vector from the origin of the w frame

$$d^w = \begin{bmatrix} t^w_{s'} \\ \operatorname{arctan2}(R^w_{s'_{32}}, R^w_{s'_{33}}) \\ -\arcsin(R^w_{s'_{31}}) \\ \operatorname{arctan2}(R^w_{s'_{21}}, R^w_{s'_{11}}) \end{bmatrix}$$

Distance to WGRs: $d(q_s, WGR)$



- use forward kinematics to get the position of the end effector at this configuration T_s^0
- get the pose of the grasp location in world coordinates

$$T_{s'}^0 = T_s^0 (T_e^w)^{-1}$$

- convert this pose from world coordinates to the coordinates of w

$$T_{s'}^w = (T_w^0)^{-1} T_{s'}^0$$

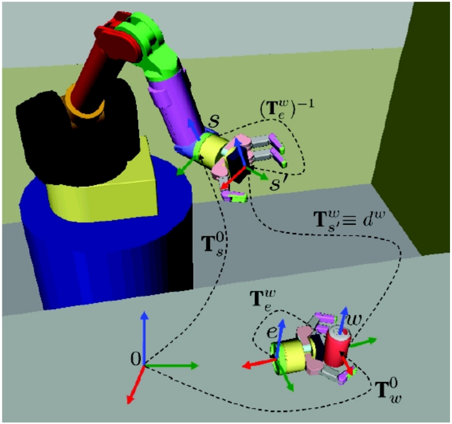
- convert $T_{s'}^w$ into a 6×1 displacement vector from the origin of the w frame

$$d^w = \begin{bmatrix} t_{s'}^w \\ \mathrm{arctan2}(R_{s'_{32}}^w, R_{s'_{33}}^w) \\ -\mathrm{arcsin}(R_{s'_{31}}^w) \\ \mathrm{arctan2}(R_{s'_{21}}^w, R_{s'_{11}}^w) \end{bmatrix}$$

- take into account the bounds B^w to get the 6×1 displacement vector Δx from d^w

$$\Delta x_i = \begin{cases} d_i^w - B_{i,1}^w & \text{if } d_i^w < B_{i,1}^w \\ d_i^w - B_{i,2}^w & \text{if } d_i^w > B_{i,2}^w \\ 0 & \text{otherwise} \end{cases}$$

Distance to WGRs: $d(q_s, WGR)$



- use forward kinematics to get the position of the end effector at this configuration T_s^0
- get the pose of the grasp location in world coordinates

$$T_{s'}^0 = T_s^0 (T_e^w)^{-1}$$

- convert this pose from world coordinates to the coordinates of w

$$T_{s'}^w = (T_w^0)^{-1} T_{s'}^0$$

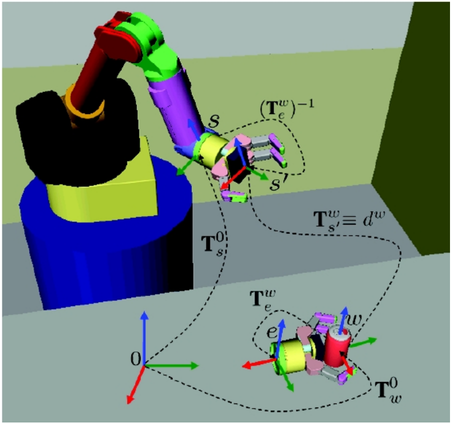
- convert $T_{s'}^w$ into a 6×1 displacement vector from the origin of the w frame

$$d^w = \begin{bmatrix} t_{s'}^w \\ \mathrm{arctan2}(R_{s'_{32}}^w, R_{s'_{33}}^w) \\ -\mathrm{arcsin}(R_{s'_{31}}^w) \\ \mathrm{arctan2}(R_{s'_{21}}^w, R_{s'_{11}}^w) \end{bmatrix}$$

- take into account the bounds B^w to get the 6×1 displacement vector Δx from d^w

$$\Delta x_i = \begin{cases} d_i^w - B_{i,1}^w & \text{if } d_i^w < B_{i,1}^w \\ d_i^w - B_{i,2}^w & \text{if } d_i^w > B_{i,2}^w \\ 0 & \text{otherwise} \end{cases}$$

Distance to WGRs: $d(q_s, WGR)$



$$d(q_s, WGR) = ||\Delta x||$$

Sampling from a WGR

Sampling from a WGR

- $d^w_{\mathrm{sample}} \leftarrow$ sample a random value between each of the bounds defined by B^w with uniform probability

Sampling from a WGR

- $d^w_{\mathrm{sample}} \leftarrow$ sample a random value between each of the bounds defined by B^w with uniform probability
- convert d^w_{sample} into a transformation matrix T^w_{sample}

Sampling from a WGR

- $d^w_{\text{sample}} \leftarrow$ sample a random value between each of the bounds defined by B^w with uniform probability
- convert d^w_{sample} into a transformation matrix T^w_{sample}
- apply the end-effector transformation to convert T^w_{sample} into world coordinates, i.e.,

$$T^0_w T^w_{\text{sample}} T^w_e$$

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q)$; $T_b.\textsc{Init}(\texttt{NULL})$

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

T_a.Init(q); T_b.Init(NULL)
while TimeRemaining() **do**

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q)$; $T_b.\textsc{Init}(\texttt{NULL})$
while $\textsc{TimeRemaining}()$ **do**
    $T_{\mathrm{goal}} \leftarrow \textsc{GetBackwardTree}(T_a, T_b)$

$T_a.\textsc{Init}(q)$; $T_b.\textsc{Init}(\texttt{NULL})$
while $\textsc{TimeRemaining}()$ **do**
  $T_{\text{goal}} \leftarrow \textsc{GetBackwardTree}(T_a, T_b)$
  if $T_{\text{goal}}.size() = 0$ or $rand(0, 1) < P_{\text{sample}}$ **then**
    $\textsc{AddIKSolutions}(T_{\text{goal}})$

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q)$; $T_b.\textsc{Init}(\texttt{NULL})$
while $\textsc{TimeRemaining}()$ **do**
  $T_{\text{goal}} \leftarrow \textsc{GetBackwardTree}(T_a, T_b)$
  if $T_{\text{goal}}.\mathit{size}() = 0$ or $\mathit{rand}(0, 1) < P_{\text{sample}}$ **then**
    $\textsc{AddIKSolutions}(T_{\text{goal}})$
  else
    $q_{\text{rand}} \leftarrow \textsc{RandConfig}()$

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q)$; $T_b.\textsc{Init}(\texttt{NULL})$
while $\textsc{TimeRemaining}()$ **do**
    $T_{\rm goal} \leftarrow \textsc{GetBackwardTree}(T_a, T_b)$
    if $T_{\rm goal}.size() = 0$ or $rand(0, 1) < P_{\rm sample}$ **then**
        $\textsc{AddIKSolutions}(T_{\rm goal})$
    else
        $q_{\rm rand} \leftarrow \textsc{RandConfig}()$
        $q^a_{\rm near} \leftarrow \textsc{NearestNeighbor}(T_a, q_{\rm rand})$

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q)$; $T_b.\textsc{Init}(\texttt{NULL})$
while $\textsc{TimeRemaining}()$ **do**
    $T_{\mathrm{goal}} \leftarrow \textsc{GetBackwardTree}(T_a, T_b)$
    if $T_{\mathrm{goal}}.size() = 0$ or $rand(0,1) < P_{\mathrm{sample}}$ **then**
        $\textsc{AddIKSolutions}(T_{\mathrm{goal}})$
    else
        $q_{\mathrm{rand}} \leftarrow \textsc{RandConfig}()$
        $q^a_{\mathrm{near}} \leftarrow \textsc{NearestNeighbor}(T_a, q_{\mathrm{rand}})$
        $q^a_{\mathrm{reached}} \leftarrow \textsc{ExtendTree}(T_a, q^a_{\mathrm{near}}, q_{\mathrm{rand}})$

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q);\; T_b.\textsc{Init}(\texttt{NULL})$

while $\textsc{TimeRemaining}()$ **do**

  $T_{\text{goal}} \leftarrow \textsc{GetBackwardTree}(T_a, T_b)$

  if $T_{\text{goal}}.\mathit{size}() = 0$ or $\mathit{rand}(0, 1) < P_{\text{sample}}$ **then**

    $\textsc{AddIKSolutions}(T_{\text{goal}})$

  else

    $q_{\text{rand}} \leftarrow \textsc{RandConfig}()$

    $q^a_{\text{near}} \leftarrow \textsc{NearestNeighbor}(T_a, q_{\text{rand}})$

    $q^a_{\text{reached}} \leftarrow \textsc{ExtendTree}(T_a, q^a_{\text{near}}, q_{\text{rand}})$

    $q^b_{\text{near}} \leftarrow \textsc{NearestNeighbor}(T_b, q_{\text{rand}})$

    $q^b_{\text{reached}} \leftarrow \textsc{ExtendTree}(T_b, q^b_{\text{near}}, q_{\text{rand}})$

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q)$; $T_b.\textsc{Init}(\texttt{NULL})$
while $\textsc{TimeRemaining}()$ **do**
  $T_{\text{goal}} \leftarrow \textsc{GetBackwardTree}(T_a, T_b)$
  if $T_{\text{goal}}.size() = 0$ or $rand(0,1) < P_{\text{sample}}$ **then**
    $\textsc{AddIKSolutions}(T_{\text{goal}})$
  else
    $q_{\text{rand}} \leftarrow \textsc{RandConfig}()$

    $q^a_{\text{near}} \leftarrow \textsc{NearestNeighbor}(T_a, q_{\text{rand}})$
    $q^a_{\text{reached}} \leftarrow \textsc{ExtendTree}(T_a, q^a_{\text{near}}, q_{\text{rand}})$

    $q^b_{\text{near}} \leftarrow \textsc{NearestNeighbor}(T_b, q_{\text{rand}})$
    $q^b_{\text{reached}} \leftarrow \textsc{ExtendTree}(T_b, q^b_{\text{near}}, q_{\text{rand}})$

    if $q^a_{\text{reached}} = q^b_{\text{reached}}$ **then**
      return $\textsc{ExtractPath}(T_a, q^a_{\text{reached}}, T_b, q^b_{\text{reached}})$

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q)$; $T_b.\textsc{Init}(\texttt{NULL})$
while $\textsc{TimeRemaining}()$ **do**
    $T_{\mathrm{goal}} \leftarrow \textsc{GetBackwardTree}(T_a, T_b)$
    if $T_{\mathrm{goal}}.\mathit{size}() = 0$ or $\mathit{rand}(0,1) < P_{\mathrm{sample}}$ **then**
        $\textsc{AddIKSolutions}(T_{\mathrm{goal}})$
    else
        $q_{\mathrm{rand}} \leftarrow \textsc{RandConfig}()$
        $q^a_{\mathrm{near}} \leftarrow \textsc{NearestNeighbor}(T_a, q_{\mathrm{rand}})$
        $q^a_{\mathrm{reached}} \leftarrow \textsc{ExtendTree}(T_a, q^a_{\mathrm{near}}, q_{\mathrm{rand}})$
        $q^b_{\mathrm{near}} \leftarrow \textsc{NearestNeighbor}(T_b, q_{\mathrm{rand}})$
        $q^b_{\mathrm{reached}} \leftarrow \textsc{ExtendTree}(T_b, q^b_{\mathrm{near}}, q_{\mathrm{rand}})$
        if $q^a_{\mathrm{reached}} = q^b_{\mathrm{reached}}$ **then**
            return $\textsc{ExtractPath}(T_a, q^a_{\mathrm{reached}}, T_b, q^b_{\mathrm{reached}})$
        else
            $\textsc{Swap}(T_a, T_b)$