CS 689 – Robot Motion Planning

Manipulation Planning

Amarda Shehu

Department of Computer Science
George Mason University

What is Manipulation Planning?

[movie: industrial]

[movie: L-shape]

[movie: industrial]

[movie: L-shape]

What is a manipulator?

[movie: industrial]

[movie: L-shape]

What is a manipulator?

- Body: articulated chain (what are configuration parameters)?
- Tool/grasper/end-effector (what are configuration parameters)?

[movie: industrial]

[movie: L-shape]

What is a manipulator?

- Body: articulated chain (what are configuration parameters)?
- Tool/grasper/end-effector (what are configuration parameters)?

How is manipulation planning a motion planning problem?

- What moves where?
- Workspace?
- Configuration space?

[movie: industrial]

[movie: L-shape]

What is a manipulator?

- Body: articulated chain (what are configuration parameters)?
- Tool/grasper/end-effector (what are configuration parameters)?

How is manipulation planning a motion planning problem?

- What moves where?
- Workspace?
- Configuration space?
  - Need to keep track of ? and ? moving in workspace?

Problem Formulation

Given:

- 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

Objective:

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

Some 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?

Two Representative Approaches

PRM-based: Nielsen and Kavraki, IROS 2000.

- Expands roadmap/graph to manipulation graph.
- Assumes stable robot grasps and object placements pre-computed and provided ahead of time.

RRT-based: Berenson et al., ICRA 2009.

- Approaches it as an inverse kinematics problem.
- Enriches any provided object placements with more and computes new robot grasps.

Assumed: stable object placements necessitating re-grasping provided ahead of time.

Assumed: stable object placements necessitating re-grasping provided ahead of time.

- How can they be pre-computed?

Assumed: stable robot grasps of given object placements provided ahead of time.

Assumed: stable object placements necessitating re-grasping provided ahead of time.

- How can they be pre-computed?

Assumed: stable robot grasps of given object placements provided ahead of time.

- How can they be pre-computed?

Focus: efficient construction of manipulation graph.

Assumed: stable object placements necessitating re-grasping provided ahead of time.

- How can they be pre-computed?

Assumed: stable robot grasps of given object placements provided ahead of time.

- How can they be pre-computed?

Focus: efficient construction of manipulation graph.

- Observation on whether motion of robot is with object grasped or not.

Observations

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

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

Manipulation Graph: Vertices

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
  - Provided or pre-computed before construction of graph
- 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
  - Provided before construction of graph
- 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
  - Focus of this approach

Transfer edge: Robot moves with object grasped by tool. What is changing?

Transfer edge: Robot moves with object grasped by tool. What is changing?

- Is robot moving in space?
- Is object moving in space?
- Is tool/grasper moving in space?

Transfer edge: Robot moves with object grasped by tool. What is changing?

- Is robot moving in space?
- Is object moving in space?
- Is tool/grasper moving in space?

An edge $\big((q^i_{\mathrm{obj}}, g, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g, q^j_{\mathrm{rob}})\big)$ indicates a tranfer (local) 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 $(q^j_{\mathrm{obj}}, q^j_{\mathrm{rob}})$

Transit edge: Robot moves to reposition its end effector/tool for object on ground. What is changing?

Transit edge: Robot moves to reposition its end effector/tool for object on ground. What is changing?

- Is robot moving in space?
- Is object moving in space?
- Is tool/grasper moving in space?

Manipulation Graph: Edges

Transit edge: Robot moves to reposition its end effector/tool for object on ground. What is changing?

- Is robot moving in space?
- Is object moving in space?
- Is tool/grasper moving in space?

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 (local) 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 $(g^j, q^j_{\mathrm{rob}})$

PRM Approach

- Node Generation:

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

PRM Approach

- Node Generation:

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

  How is sampling done for each of the components of the configuration?

PRM Approach

- Node Generation:

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

  How is sampling done for each of the components of the configuration?

- 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:

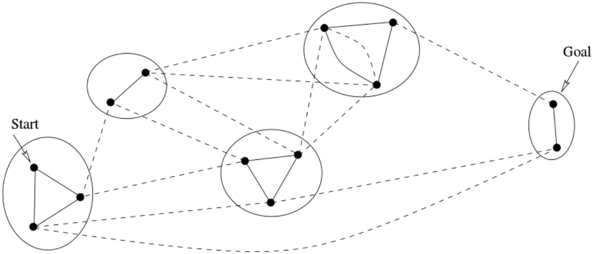
  for $i = 1, \ldots, N$ **do** sample a node $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$

  How is sampling done for each of the components of the configuration?

- Edge Generation:

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

  How is local path generated for transfer or transit edge?



Solid lines represent transit paths, and dotted lines represent transfer paths.

Challenges:

- 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)

Challenges:

- 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
- From the people that gave you the Lazy PRM...

[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

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

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

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

Query Stage

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

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

[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

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

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 $prob(e) \leftarrow 0.9999$
    **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**
                $prob(e) \leftarrow 1 - \frac{time(e)}{total\_time}$
```

A two-level Fuzzy PRM for Manipulation Planning

[Nielsen, Kavraki: IROS 2000]

Manipulation Graph

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

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:    $prob(q, q') \leftarrow P^*(l)$

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
                $prob(e) \leftarrow 1 - \frac{time(e)}{total\_time}$

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**

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:                 $prob(e) \leftarrow 1 - \frac{time(e)}{total\_time}$

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

[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
                $prob(e) \leftarrow 1 - \frac{time(e)}{total\_time}$

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

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:                 $prob(e) \leftarrow 1 - \frac{time(e)}{total\_time}$

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 $\ell(q', q'')$

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: $prob(e) \leftarrow 1 - \frac{time(e)}{total\_time}$

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`

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
                $prob(e) \leftarrow 1 - \frac{time(e)}{total\_time}$

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`
            else
                update $prob(q', q'')$ based on collision resolution $\ell(q', q'')$

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
        $prob(e) \leftarrow 1 - \frac{time(e)}{total\_time}$

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`
      else
        update $prob(q', q'')$ 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

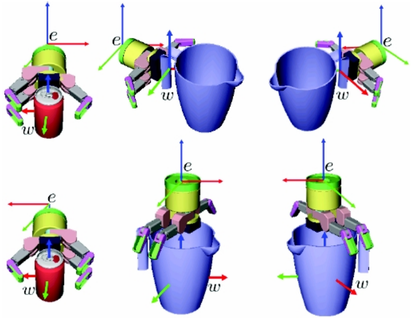
[Berenson, Srinivasa, Ferguson, Collet, Kuffner: ICRA 2009]

- 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 reachable, it may take the planner a long time to find solutions to these goal configurations

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

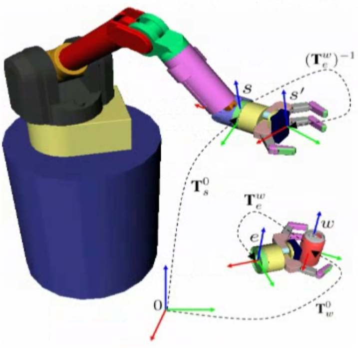
Proposed Approach

- 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



- Two WGRs describe grasping a soda can
- Bounds allow rotation around z axis of w

- Reference frame w attached at object specifying pre-computed *grasp pose*
- Workspace bounds B^w specifying flexibility around target grasp w: $[(x_{\min}, x_{\max}), (y_{\min}, y_{\max}), (z_{\min}, z_{\max}), (\psi_{\min}, \psi_{\max}), (\theta_{\min}, \theta_{\max}), (\phi_{\min}, \phi_{\max})]$



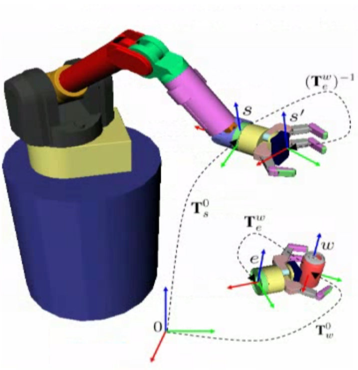
- To allow offset for end-effector, transform T_e^w specifies end-effector pose relative to the (w) reference frame of the desired grasp
- Simple operations can be done: $T_w^0 T_e^w$ now specifies a target pose of end effector in world coordinate frame
- One can sample alternative pose for end effector from B^w, and then convert to world coordinate frame to provide an end-effector goal pose to IK solver

Why Use WGRs for Manipulation Planning

- Sampling from B^w (in the provided range for each of the 6 coordinates that specify the pose of target, pre-specified grasp) gives alternative grasper pose in (w/object's) coordinate frame.
- Sample can be converted into new, sampled goal pose for end-effector.
- IK can be used to steer manipulator towards sampled goal end-effector pose.
- All encapsulated in an IK bi-directional RRT (IKBiRRT) so as to deal with the usual get-stuck (subptimal) behavior of gradient-descent type methods for IK.
- A distance measure can be specified to give a sense of how far or near two end-effector configurations are for RRT.

Some Sampling Target End-Effector Pose from 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}, which specifies the *sampled grasper pose* relative to the coordinate frame w of the target grasp.
- convert the *sampled grasper pose* into a *sampled pose for the end-effector*, still in the coordinate frame of w (target grasp pose)



$$T^w_{\text{sample}} \cdot T^w_e$$

- convert the sampled end-effector pose in world coordinates

$$T^0_{\text{sample}'} = T^0_w T^w_{\text{sample}} T^w_e$$

- $T^0_{\text{sample}'}$ is passed to an IK solver to generate solution(a)s, which are checked for collisions. Only collision-free solutions are added to the RRT.

Distance Measurement for RRT

- use FK to get end-effector pose at current q_s configuration: T^0_s is pose of end-effector in world coordinates.
- get pose of grasp, if object held there, in world coordinates

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

- convert it from world to coordinates of w

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

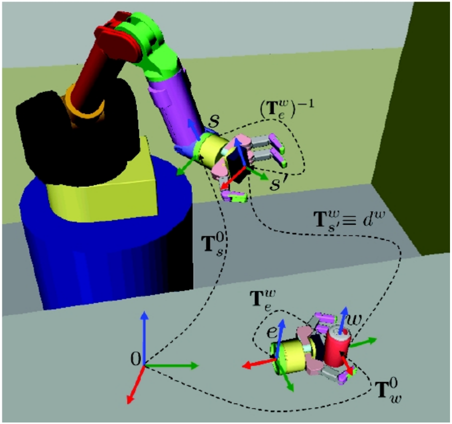
- convert $T^w_{s'}$ into a 6×1 displacement vector from origin of w frame

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

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

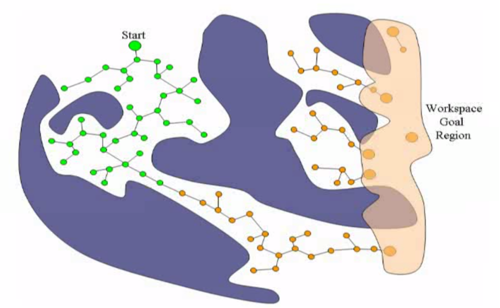
$$\Delta x_i = \begin{cases} d^w_i - B^w_{i,1} & \text{if } d^w_i < B^w_{i,1} \\ d^w_i - B^w_{i,2} & \text{if } d^w_i > B^w_{i,2} \\ 0 & \text{otherwise} \end{cases}$$

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



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

- Grows one tree from start and one tree from goal configuration.
- At each iteration chooses between one of two modes: exploration through standard BiRRT and sampling from the set of WGRs W. The probability of choosing the mode is controlled by the parameter P_{sample}.
- Goal configurations sampled from a WGR are injected into the backwards tree that grows from goal.
- Termination when both trees meet at some configuration.



Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q_s)$; $T_b.\textsc{Init}(\texttt{NULL})$
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 $\textsc{ExtractPath}(T_a, q^a_{\text{reached}}, T_b, q^b_{\text{reached}})$
    else
      $\textsc{Swap}(T_a, T_b)$

[movie]