CS 485 – Autonomous Robotics

Manipulation Planning

Amarda Shehu

Department of Computer Science
George Mason University

What is Manipulation Planning?

[movie: industrial]

[movie: L-shape]

What is Manipulation Planning?

[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?

What is Manipulation Planning?

[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?

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 *avoding collisions*

- 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?

Manipulation Graph: Edges

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?

Manipulation Graph: Edges

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 $\left((q^i_{\mathrm{obj}}, g, q^i_{\mathrm{rob}}), (q^j_{\mathrm{obj}}, g, q^j_{\mathrm{rob}})\right)$ 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?

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 $(q_{\mathrm{obj}}^{i}, g^{i}, q_{\mathrm{rob}}^{i})$

  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:

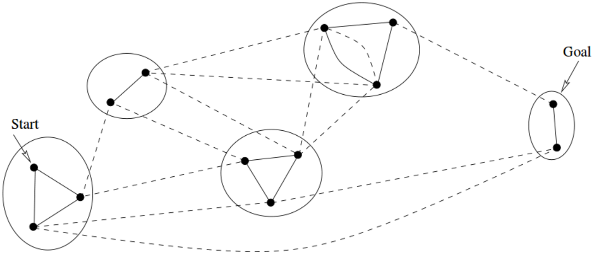
  for $i = 1, \ldots, N$ **do** sample a node $(q^i_{\mathrm{obj}}, g^i, q^i_{\mathrm{rob}})$

  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 and Key Idea

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

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

[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**

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 $prob(e) \leftarrow 0.9999$

[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 $prob(e) \leftarrow 0.9999$

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**

[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**

[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

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:                 $prob(e) \leftarrow 1$

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

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

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**

[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

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 $\mathit{prob}(e) \leftarrow 0.9999$
4:    **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

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** $\mathit{prob}(e) \neq 1$ **then**
5:          run low-level fuzzy PRM on e for a short period of time
6:          **if** success **then**
7:             $\mathit{prob}(e) \leftarrow 1$
8:          **else**
9:             $\mathit{prob}(e) \leftarrow \frac{1 - \mathit{time}(e)}{\mathit{total\_time}}$

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

[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 $(q^i_{\text{obj}}, g^i, q^i_{\text{rob}})$, $i = 1, \ldots, N$ of the manipulation graph
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**
    if $g^i = g^j$ **then** add e as a transfer edge and set $prob(e) \leftarrow 0.9999$
    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

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

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

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`
            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_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
```

Query Stage

```
while no solution found do
    σ ← compute most probable path in the
    manipulation graph
    for each edge e ∈ σ do
        if prob(e) ≠ 1 then
            run low-level fuzzy PRM on e for a
            short period of time
            if success then
                prob(e) ← 1
            else
                prob(e) ← (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') ← P*(l)
if mode = "QUERY" then
    φ ← compute most probable path in G_e
    repeat
        (q', q'') ← edge in φ with lowest
        probability
        if prob(q', q'') ≠ 1 then
            run subdivision collision checking to
            validate (q', q'') at resolution
            ℓ(q', q'')
            increment ℓ(q', q'')
            if collision then
                remove (q', q'') from G_e and
                return failure
            else
                update prob(q', q'') based on
                collision resolution ℓ(q', q'')
    until all edges in φ have prob 1
    return success
```

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

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

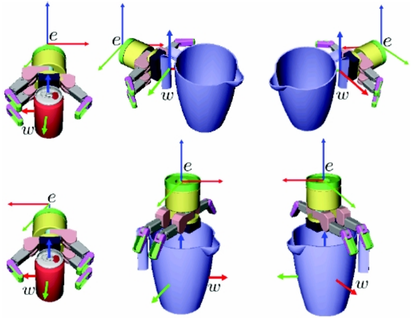
- Manipulation planners often require specification of a set of stable grasp configurations
- This forces the planner to use only these configurations as goals
- If the chosen goal configurations are unreachable, the planner will fail
- Even when reachable, it may take the planner a long time to find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

Manipulation Planning with Workspace Goal Regions

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

Proposed Approach

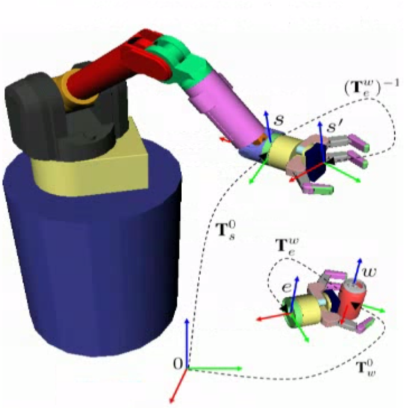
- Introduce concept of Workspace Goal Regions (WGRs)
- WGR allows the specification of continuous regions in the six-dimensional workspace of end-effector poses as goals for the planner



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

Desired properties of a WGR:

- easy to describe
- easy to sample
- easy to define distance from robot configuration to WGR

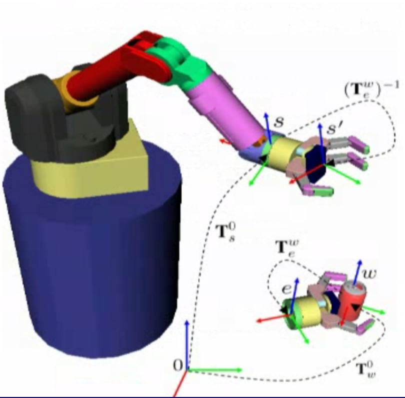


Definition of a WGR

What we need:

- A reference frame w attached at object, which specifies given, desired, pre-computed grasp pose
- An end-effector transform T_e^w that specifies the pose of the end effector relative to the (w) reference frame of the desired grasp
- Workspace bounds B^w specifying flexibility around target grasp w:

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



Why?

- We can sample bounds in the provided range for each of the 6 coordinates (pose of target, pre-specified grasp, essentially drawing samples from a WGR.
- Samples specify possible alternative grasps relative to the coordinate frame of the pre-computed target grasp.
- They can be converted into new sampled goal poses for the end-effector through the transformation T_e^w.
- IK can then be employed to steer the manipulator towards a sampled goal end-effector pose.
- A distance measure can also be specified to give a sense of how far or near two end-effector configurations are.
- All is 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.

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:

- $d^w_{\text{sample}} \leftarrow$ 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 sample relative to the coordinate frame w of the target grasp.

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}, which specifies the sample relative to the coordinate frame w of the target grasp.
- convert it into a sample for the end-effector, still in the coordinate frame of w (target grasp pose)

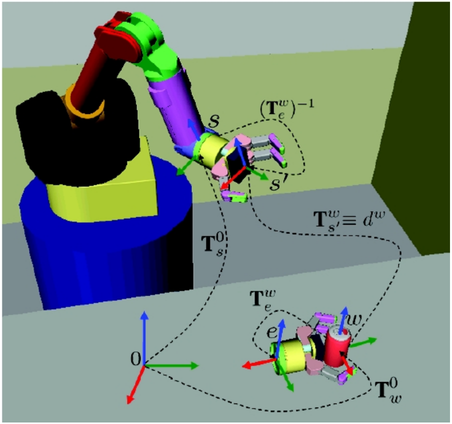
$$T^w_{\mathrm{sample}} \cdot T^w_e$$

- convert the sampled end-effector pose in world coordinates

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

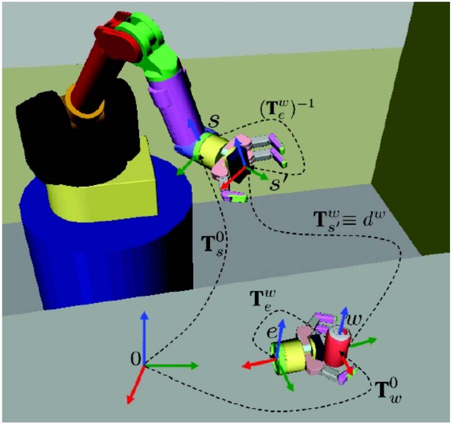
- The transform is passed to an IK solver to generate some number of solutions, which are checked for collisions. Only collision-free solutions are added to tree.

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



- use FK to get end-effector pose at current q_s configuration: T_s^0 is pose of end-effector in world coordinates.

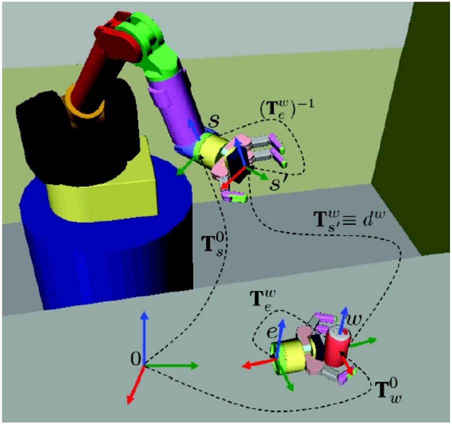
Distance to WGRs: $d(q_s, WGR)$



- 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

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

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



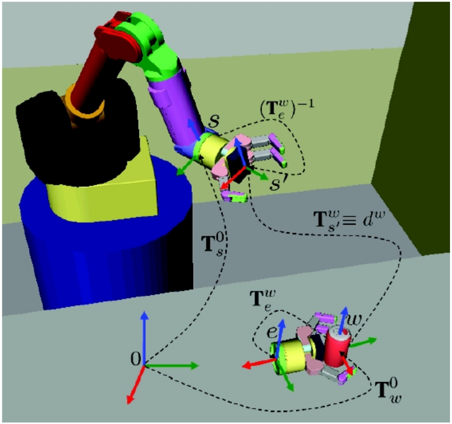
- use FK to get end-effector pose at current q_s configuration: T_s^0 is pose of end-effector in world coordinates.
- get pose of grasp, if object held there, in world coordinates

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

- convert it from world to coordinates of w

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

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



Distance Measurement

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

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

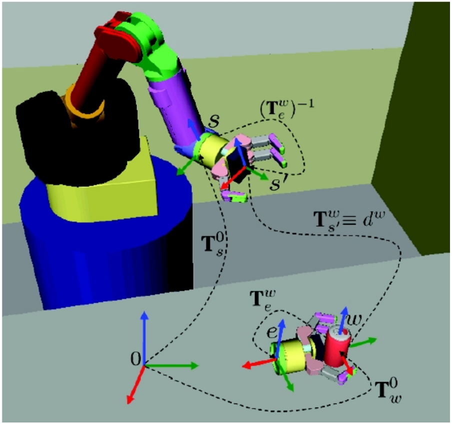
- convert it from world to coordinates of w

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

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

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

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



Distance Measurement

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

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

- convert it from world to coordinates of w

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

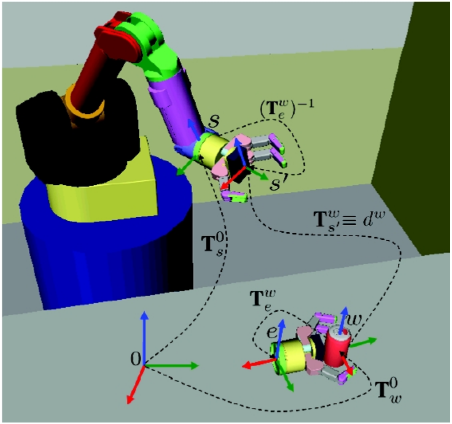
- convert $T_{s'}^w$ into a 6×1 displacement vector from origin of w frame

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

- take into account bounds B^w to get 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: $d(q_s, WGR)$



Distance Measurement

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

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

- convert it from world to coordinates of w

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

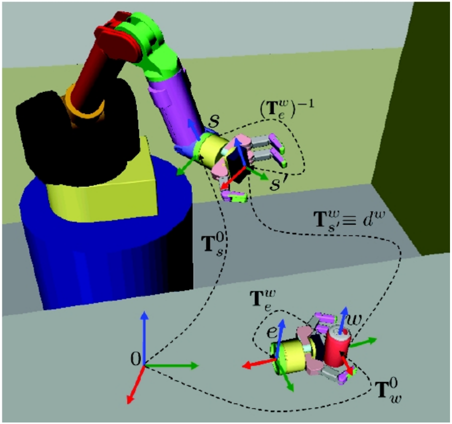
- convert $T_{s'}^w$ into a 6×1 displacement vector from origin of w frame

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

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

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

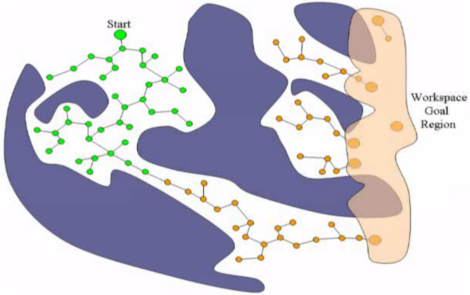
Distance to WGRs: $d(q_s, WGR)$



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

Inverse Kinematics Bi-Directional RRT (IKBiRRT): Overall Approach

- 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)

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

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

T_a.INIT(q_s); T_b.INIT(`NULL`)
while TIMEREMAINING() **do**

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

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

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

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 $T_{\text{goal}}.\mathit{size}() = 0$ 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)

$T_a.\textsc{Init}(q_s)$; $T_b.\textsc{Init}(\texttt{NULL})$
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
        $q_{\text{rand}} \leftarrow \textsc{RandConfig}()$
        $q^a_{\text{near}} \leftarrow \textsc{NearestNeighbor}(T_a, q_{\text{rand}})$

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

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

Inverse Kinematics Bi-Directional RRT (IKBiRRT)

$T_a.\textsc{Init}(q_s)$; $T_b.\textsc{Init}(\texttt{NULL})$
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
        $q_{\mathrm{rand}} \leftarrow \textsc{RandConfig}()$

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

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

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

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]