Humanoid Motion Planning for Dual-Arm Manipulation and Re-Grasping Tasks

Researchers: Nikolaus Vahrenkamp, Dmitry Berenson, Tamim Asfour, James Kuffner, Rudiger Dillmann

Presented by Jared Greene
February 7, 2017
Grasping is needed for non-trivial integration of humanoid robotics into our world
  ○ Learning new tasks or adapting known tasks often require the use of new tools or interactions with new objects

This is a difficult problem
  ○ Unpredictability of the environment and poses
  ○ Collisions
  ○ Coordination (dual grasping)

Overview of Grasping and Grasp Planning

1. Identify a target grasp
2. Find target pose
3. Solve for joint configs
4. Plan a valid path
Grasping is needed for non-trivial integration of humanoid robotics into our world

- Learning new tasks or adapting known tasks often require the use of new tools or interactions with new objects

This is a difficult problem

- Unpredictability of the environment and poses
- Collisions
- Coordination (dual grasping)

Overview of Grasping and Grasp Planning

1. Identify a target grasp
2. Find target pose
3. Solve for joint configs
4. Plan a valid path
Single Arm Motion Planning

- Predefined grasp configuration
- $J^+$-RRT
- Inverse Kinematics RRT
Single Arm Planning - General tactic

- Solve for both the joint configuration path planning simultaneously

- This avoids conflicts
  - Invalid config for the environment (if kinematics are determined first)
  - Invalid actuator configuration for the hardware (if planning is done first)
Single Arm Planning - Predefined grasp configuration

- Inverse Kinematics Solver run for every pose in the database
- Can fail to find a valid configurations for a given pose
  - Infinite number of IK solutions
  - Finite number of solutions would be saved
  - Could have only saved solutions that are invalid in the given context
- Expensive - especially when feasible grasps that are stored in the database are densely sampled
Single Arm Planning - $J^+$-RRT

- Based on the RRT-JT
- Moves towards a series of potential goals (set of feasible grasps)
- Uses the pseudoinverse to compute goal directed expansion steps
- Does not require an IK-solver
- Extends to goal with some probability for each trial
Single Arm Planning - Inverse Kinematics RRT

- Bi-directional RRT
- Grows from start immediately
- Grows from goal once an IK has been found and added
- Add a new goal (call the IK solver) with some probability
- Grows tree and tries to connect to both start and goal
  - Success if both connect

```c
Algorithm 3: IK-RRT(q_{start}, p_{goal})
1. RRT1.AddConfiguration(q_{start});
2. RRT2.Clear();
3. while (!TimeOut()) do
   4. if (#IKSolutions == 0 || rand() < p_{IK}) then
      5. grasp ← GetRandomGrasp(gc);
      6. p_{target} ← ComputeTargetPose(p_{obj}, grasp);
      7. q_{IK} ← ComputeIK(p_{target});
      8. if (!Collision(q_{IK})) then
         9. RRT2.AddConfiguration(q_{IK});
      else
         10. q_{r} ← GetRandomConfiguration();
         11. if
            12. (RRT1.Connect(q_{r}) & RRT2.Connect(q_{r}))
            then
               13. Solution ← BuildSolutionPath(q_{r});
               14. return PrunePath(Solution);
```
### Average Runtimes for 100 single-arm reaching trials

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg Runtime Without Obstacle</th>
<th>Average Runtime with Obstacle</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J^+\text{-RRT}$</td>
<td>2032 ms</td>
<td>18390 ms</td>
</tr>
<tr>
<td>IK-RRT</td>
<td>140 ms</td>
<td>480 ms</td>
</tr>
</tbody>
</table>

**Significant improvement over $J^+\text{-RRT}$**
Dual-Arm Motion Planning

- Flexible Grasping
- Dual-arm re-grasping
## Dual-arm planning - Added complexities

### Single-Arm

- **Tasks**
  - Choose grasp
  - Determine pose
  - Find configuration
  - Find path

- **Collision concerns**
  - Environment, robot, object

- **Degrees of freedom**
  - 10 (including the hip)

### Dual-Arm

- **Tasks**
  - Choose grasp of non-grasping hand (ngh)
  - Determine pose of ngh
  - Determine pose of grasping hand
  - Determine configuration for ngh
  - Determine configuration for grasping hand
  - Determine path for ngh
  - Determine path for grasping hand

- **Collision concerns**
  - Environment, robot, object (now in motion), other arm

- **Degrees of freedom**
  - 23 (flexible)
  - 17 (fixed)
Dual-arm planning - Re-grasping with $J^+$-RRT & IK-RRT

**$J^+$-RRT**

- More reminiscent of bidirectional-RRT
- Extend to goal now tries to move both arms towards the closest point between the two of them

**IK-RRT**

- IK solver now operates on grasps rather than poses
- Object is viewed as a fixed attachment of the grasping arm

### Average Runtimes for 100 Dual-Arm Re-Grasping Planners

<table>
<thead>
<tr>
<th></th>
<th>Avg Runtime Without Obstacle</th>
<th>Average Runtime with Obstacle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual Arm with $J^+$-RRT</td>
<td>1662 ms</td>
<td>5192 ms</td>
</tr>
<tr>
<td>Dual Arm with IK-RRT</td>
<td>278 ms</td>
<td>469 ms</td>
</tr>
</tbody>
</table>
Single Arm Inverse Kinematics

- Random Sampling
- Reachability Space
- Gradient Descent
• Inverse Kinematic Solver
  ○ As opposed to a forward kinematic approach
  ○ Very expensive

• Want to use IK sparingly
Single Arm Inverse Kinematics - Random Sampling

- Random sampling of free parameters
  - 3 DoF in the hips are randomly sampled
  - Increases the reachability space
  - Shrinks the computational costs by reducing the complexities for IK solvers and path planners
Single Arm Inverse Kinematics - Reachability Space

● Issue
  ○ IK attempts are expensive
  ○ Which should we attempt?

● Reachability Space
  ○ Represent reachability as a space of 6D voxels
  ○ Each voxel contains information about the probability that an IK attempt will succeed

● Use
  ○ Once an IK attempt is requested on a pose, only attempt if the voxel that corresponds to it has sufficient probability
  ○ Prevents attempts on poses which are out of reach
Once an IK attempt is requested on a valid voxel the IK will execute
  ○ But we want to make sure we have the greatest chance of succeeding

Gradient Descent
  ○ Pick the adjacent voxel with greatest likelihood (must also be greater than current voxel)
  ○ Local maximum likelihood of success
  ○ Gradient descent with 1 voxel step-size
Reachability Space for ARMAR-III

2D projection of Gradient Descent Optimization
### 100 IK-queries for a 10 DoF IK Solver

<table>
<thead>
<tr>
<th></th>
<th>Avg Runtime Without Obstacle</th>
<th># of IK Calls</th>
<th>Average Runtime with Obstacle</th>
<th># of IK Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without R-Space</td>
<td>1404 ms</td>
<td>101.9</td>
<td>2880 ms</td>
<td>217.3</td>
</tr>
<tr>
<td>With R-Space</td>
<td>60 ms</td>
<td>6.1</td>
<td>144 ms</td>
<td>13.6</td>
</tr>
</tbody>
</table>

Large improvement with reachability space
Dual-Arm Inverse Kinematics

- Changes & Performance
### Dual Arm IK-Solver - Flexible

- 23 DoF (7, 7, 6, 3)

### Dual Arm IK-Solver - Grasped with One

- 17 DoF (7, 7, 3)

### 100 IK-Queries for Dual-Arm IK Solvers

<table>
<thead>
<tr>
<th></th>
<th>Avg Runtime Without Obstacle</th>
<th># of IK Calls</th>
<th>Average Runtime with Obstacle</th>
<th># of IK Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible Grasp</td>
<td>47 ms</td>
<td>3.3</td>
<td>161</td>
<td>6.5</td>
</tr>
<tr>
<td>Grasped w/ One</td>
<td>162 ms</td>
<td>3.2</td>
<td>220 ms</td>
<td>4.3</td>
</tr>
</tbody>
</table>
Dual Arm Grasping - Bowl

Dual-Arm Re-Grasping - Wox

Dual-Arm Re-Grasping - Plate over obstacle