Planning with hybrid knowledge in the GeRT project

F. Lagriffoul, L. Karlsson, F. Pecora and A. Saffiotti

Center of Applied Autonomous Sensor Systems (AASS)
Örebro university - Sweden
The GeRT project

Generalizing Robot manipulation Tasks
Aim: coping with novelty in manipulation tasks.

- Abstract away the variability in objects (shape, size and pose)
- Adapt to the context in which the task is performed (obstacles, initial configuration)
- Perform tasks it has not encountered before, given the necessary building blocks for that task
The planning problem in GeRT

Achieving this involves different research areas:

- **Planning**
  To compute a sequence of actions to perform the task

- **Learning**
  To extract knowledge from example programs

- **Machine perception**
  To figure out the locations and types of objects in the scene

- **Grasping**
  Which is a difficult problem on its own
The planning problem in GeRT

Achieving this involves different research areas:

- **Planning** → **Hybrid planning**
  
  To compute a sequence of actions to perform the task

- **Learning**
  
  To extract knowledge from example programs

- **Machine perception**
  
  To figure out the locations and types of objects in the scene

- **Grasping**
  
  Which is a difficult problem on its own
The planning problem in GeRT

Hybrid planning

Geometric knowledge

Symbolic knowledge

Initial state
- at robot1 p1
- at robot2 p2

Goal state
- at robot1 p1'
- at robot2 p2'

2 plans are solution to the symbolic problem:

**PLAN1:**
- move robot1 p1'
- move robot2 p2'

**PLAN2:**
- move robot2 p2'
- move robot1 p1'
The planning problem in GeRT

Hybrid planning
Relation to workshop topics

- **Terminology**: task descriptions, action recipes, skills and other primitives, and what are their relationships?
- **Conventions**: shared definitions, conventions (coordinate systems, units), and data structures
- **Scene graphs**: How should data be represented (maps, objects, actions)? What data should be annotated and how? What kind of reasoning is performed or needed?
- **Reuse of knowledge**: How can a robot decide which knowledge (e.g., map or skill) to reuse in a new situation?
- **Reuse of tools**: What existing software modules, algorithms, libraries, or APIs can be reused?
- **Knowledge engineering**: How will/should the knowledge base grow? What are the processes leading to creation of a substantial knowledge base useful in real applications, i.e. the bootstrapping of the KB?
Terminology, Action recipes

Today:

- Hand-written Python scripts

GeRT:

- Operators and methods
- Geometric knowledge

Hybrid planning
- Executable plan
- Python script

Goal

(Example programs)

Abstraction Learning
Knowledge representation

- **Symbolic Knowledge**

<table>
<thead>
<tr>
<th>States</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>On (Cup1, table)</td>
<td>Operator: Grab(hand, obj)</td>
</tr>
<tr>
<td>Holding (Teapot, RightHand)</td>
<td>Preconditions: Free(hand), Clear(obj)</td>
</tr>
<tr>
<td>Empty (Cup1)</td>
<td>Effects: NOT Free(hand), NOT On(obj, table),</td>
</tr>
<tr>
<td></td>
<td>Holding(hand, obj)</td>
</tr>
</tbody>
</table>

- **Geometric Knowledge**
  - Geometries of the robot and objects (trimeshes)
  - Position and orientation of objects (transformation matrices)
  - Configuration of the robot (vector)
  - Relative positions (transformation matrices)
Example programs
Reuse of knowledge: Abstraction

Example programs

Abstraction / Learning
Reuse of knowledge: Abstraction

Example programs

Abstraction / Learning

Operators

Methods
Reuse of knowledge: solving a new task

Operators

Methods

Hybrid planning

Goal

Geometric knowledge
Reuse of knowledge: solving a new task

Hybrid planning

Goal

Operators

Methods

Geometric knowledge

Python script:

```python
1. select_cub(tube)
2. get_palm预备 ready
3. open-right-hand
4. select_cub(tube)
5. get_palm open-right-hand
6. get_palm green-cube with right hand
7. select_cub(tube)
8. locate_object(tube, green)
9. get_palm open left-hand
10. open-left-hand
11. select_cub(common positions)
12. get_palm open left-hand
13. get_palm open cube with left hand
14. select_cub(tube)
15. locate_objects(tube, left)
16. get_palm left-preparing-rod
17. get_palm green-cube
18. select_cub(common positions)
19. get_palm right-hand-prep-cup
20. app_gear_interface-bind(rightside", "tube-green")
21. 900
22. select_cub(rightside)
23. get_palm right-hand above right
24. get_palm green-cube at right
25. get_palm green-cube at right
26. select_cub(common positions)
27. get_palm right-hand-prep-cup
28. app_gear_interface-bind(rightside", "tube-green")
29. 900
30. select_cub(rightside)
31. get_palm right-hand above right
32. get_palm right-hand above right
33. get_palm right-hand above right
34. get_palm right-hand above right
35. get_palm right-hand above right
36. get_palm right-hand above right
37. select_cub(common positions)
38. get_palm open-right-hand
39. app_gear_interface-bind(rightside", "tube-green")
40. select_cub(tube)
41. get_palm right-hand above right
42. get_palm right-hand above right
43. app_gear_interface-bind(rightside", "tube-green")
44. select_cub(tube)
45. get_palm right-hand above right
46. app_gear_interface-bind(rightside", "tube-green")
47. select_cub(common positions)
48. get_palm left-hand-prep-cup
49. app_gear_interface-bind(leftside", "tube-left")
50. 900
51. select_cub(leftside)
52. get_palm left-hand above left
53. get_palm left-hand above left
54. get_palm left-hand above left
55. get_palm left-hand above left
56. get_palm left-hand above left
57. select_cub(common positions)
58. get_palm open-left-hand
59. app_gear_interface-bind(leftside", "tube-left")
60. select_cub(tube)
61. get_palm open-left-hand
62. app_gear_interface-bind(leftside", "tube-left")
63. left-arm-away
64. get_palm left-arm-away
```
The End

Thanks for your attention.

Questions?