On DRC Hose Installment and Related

Task, Approach, Status

Hao Dang, Peter K. Allen, Columbia University

Youngbum Jun, Paul Oh, Drexel University

Hose Installment Task Specification

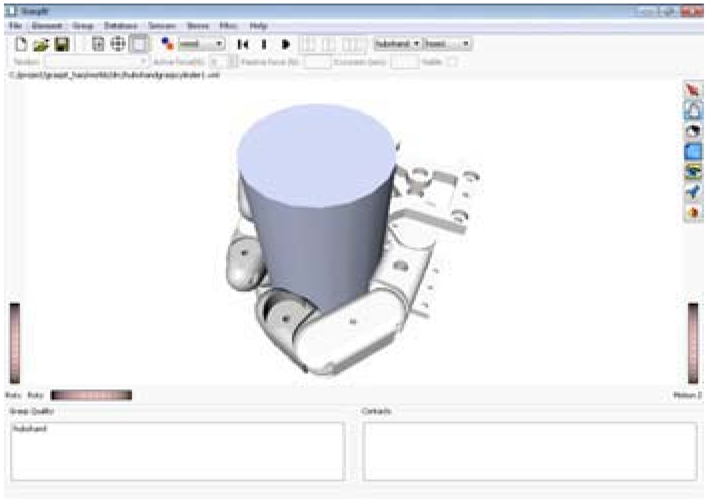
- Insert a hose into a hydrant
- Five sub-tasks to complete a hose installment task
  - Detect objects – hose and hydrant, etc
  - Grasp the hose
  - Lift up the hose
  - Move to the hydrant
  - Insert into the hydrant

Our Approach

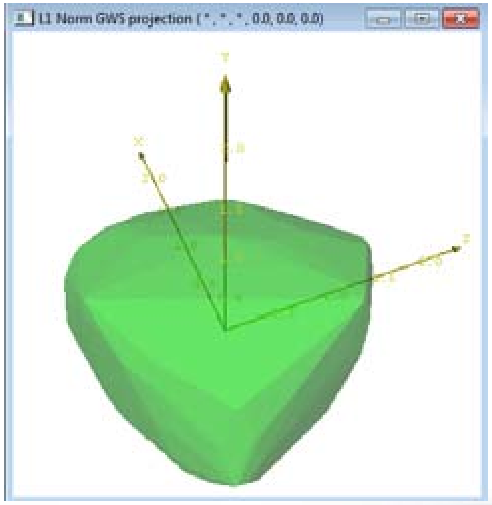
- Three major steps – reconstruction + planning + execution
  - Perception
    - recognize the objects – hose, hydrant, floor, table, obstacles...
    - reconstruct the scene for the planning/simulation environment
  - Planning
    - Grasping
      - Store a grasp, plan a grasp
      - plan collision free grasping trajectory
    - Manipulation
      - plan collision free rotation trajectory
  - Execution
    - Executing the trajectories in physics simulatio / real robot

Grasp Planning

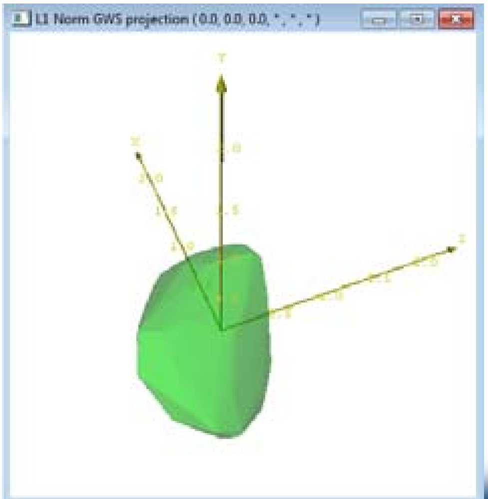
- *GraspIt!* – a universal grasp modeling, analysis, and simulation tool
  - Manually define a grasp within the UI (or automatically plan)
    - stability measurement: material + kinematics
  - Store grasps into a local/remote grasp database
    - grasp with tags – rotational symmetry
  - Retrieve grasps from a local/remote grasp database



GraspIt! GUI



GWS projected onto force space



GWS projected onto torque space

Manipulation Planning

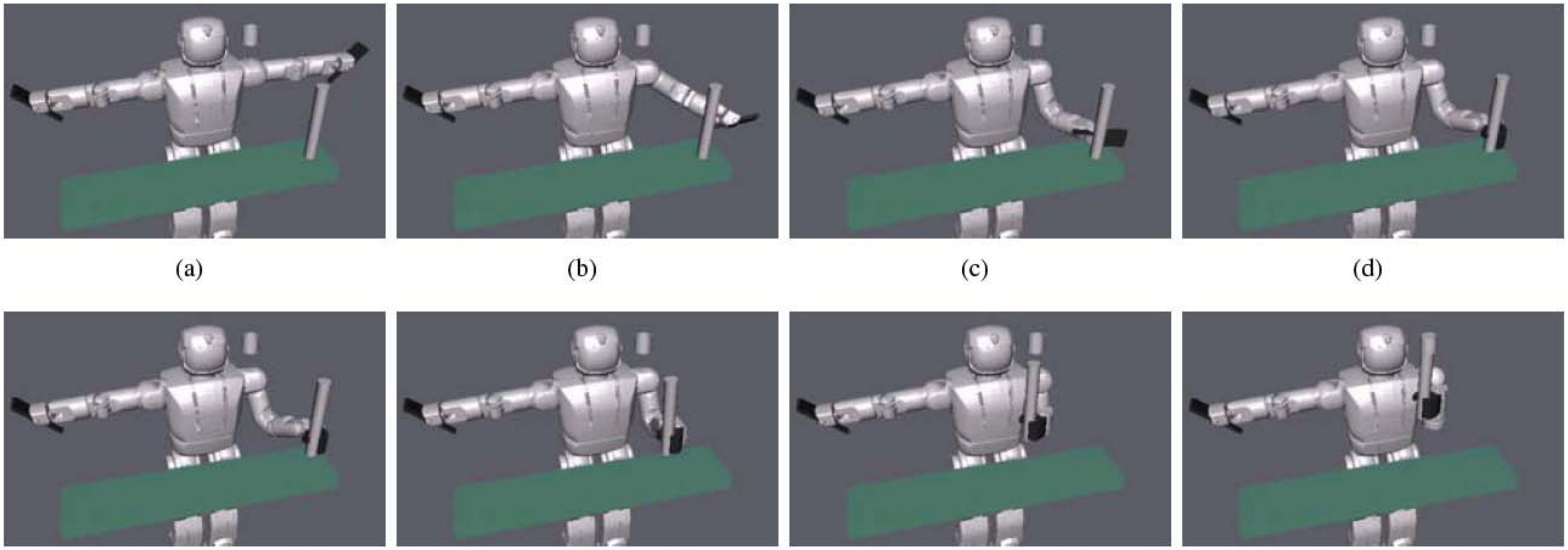
- CBiRRT + OpenRAVE
- Scene reconstructed using object models
  - Object models are pre-defined
  - Perception provides the pose of each object
- OpenRAVE uses the reconstructed scene and plans collision-free trajectories for grasping + manipulation
  - High-level trajectory is pre-determined for the task
    - lift-up, insertion, transportation

Manipulation Execution

- Simulation generates collision-free trajectories
- Physics simulation
  - Run generated trajectories inside OpenRAVE
- Execute with Hubo
  - Sample the generated collision-free trajectories
  - Convert the trajectories into Hubo format
  - Double check the velocity and acceleration along the trajectory
  - Run the trajectories on Hubo

Experiments

- Simulation experiment
  - grasp → lift up → move to hydrant → insert into hydrant

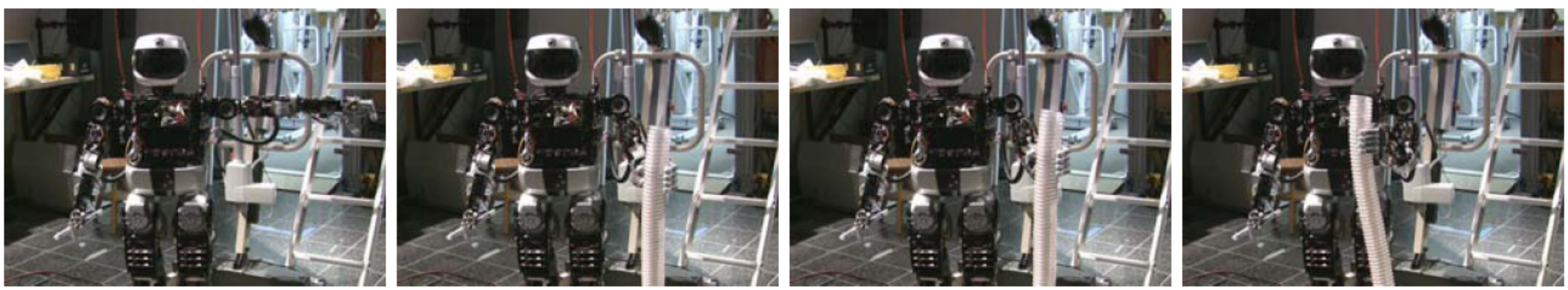


(a) (b) (c) (d)
(e) (f) (g) (h)

Experiments

- Physical experiment

(a) (b) (c) (d)



(e) (f) (g) (h)

Some Observations We've Made

- Physics simulation
  - Servo controller is not performing accurately consistently
  - Drift effects
  - Performs differently on different machines with different computational power
- Physical experiments
  - More flexible arms
  - Stronger hands

New Hand/Arm Requirements

- make the arm 7-dof instead of 6-dof (huboplus is 6-dof)
- have joint angle sensor on each joint
- have 6-dof force/torque sensor on the wrist of the hand
- strain gauges on finger joints
- attach tactile sensors to the surface of the fingers (5mm by 5mm)
- have rubber coating on the fingers
- maybe have one less finger but make the motors more powerful

What's Next

- Perception System
  - In our experiments, we used a Mocap system
  - Markers were used to indicate the locations of the objects
  - We need to integrate object recognition
- Robot control
  - Some kind of adaptive grasping
  - We need to have better API's for the execution of trajectories
  - We are currently looking at ACH
  - Compliance?

Some Experience We Learned from the ARM-S

- Calibration
  - Arm kinematics
    - Play due to cable driven mechanism
  - Camera calibration
    - Swiss ranger 4000, point gray camera, bumblebee stereo
  - Neck kinematics
    - Two pan/tilt units connected
    - Play due to gap between gears
    - Non-linearity due to springs holding back the pan/tilt units
- Unexpected Collision
  - Object padding for collision avoidance
- Reactive Grasping
  - Sensor capability: vision, force/torque, strain gauges, tactile

Acknowledgement

- Youngbum Jun
- Dmitry Berenson
- Robert Ellenberg