Physics-Based Grasping under Uncertainty

Michael Koval
mkoval@cs.cmu.edu

February 1, 2017
16-662 Robot Autonomy

Scribes:
Juan Pablo Vega (jvega)
Sanjay Narayanan (svnaraya)
Yuan Hu (yuanh)
Task: Getting HERB to grasp the FUZE bottle
Given that we can get force/form closure, we need to:

- Generate grasp
- Evaluate grasp
- Generate poses for the robot's fingers using Inverse Kinematics
- Actuate robot's manipulator to achieve desired poses

With uncertainty at every stage, the grasp will fail.
Video showing robot successfully grasping bottle
With uncertainty, one of the fingers might not attain the desired pose, resulting in a failure in grasping.
Video showing robot failing to grasp the drill.
Trying to get green object to green circle (goal region) -> standard motion planning problem
Usual Approach:
Using collision checker to generate a map -> PRM/RRT
Robots are often never good enough, even the most expensive/advanced robots.

Uncertainty will always be an issue.
Sources of uncertainty:
- Mismatch between perceived environment and real world
- Noise in measurements and perception
- Errors in actuators and controllers
- Flaws in simulators

What caused these failures?
Just a small error in perception (mismatch in what we think the environment looks like vs. what it actually looks like) can result in a failure in planning.
object pose uncertainty
object pose uncertainty
Control uncertainty: even if we know what environment looks like, robot may not move as expected.
Analogy: Measuring the odometry of a car before the transmission
Model uncertainty ->
same model in two different simulators can give different results.
Open loop in controlled environments (factories) works because there are
- Well-defined fixtures
- Allows for realignments
- Custom end-effectors (suction cups for beer bottles)
Source: https://youtu.be/S8nkaTer2_o - ESSEMTEC pick and place machine
How can we manipulate under uncertainty?
The things that work for industrial robots are not always feasible for a robot like HERB.

How do we plan algorithms/policies to make up for the uncertainty, without altering the environment?
Closed-loop -> using sensing feedback (kinect, LIDAR, depth sensors, force/torque sensors, tactile sensing)

Open -loop -> using the environment -> pushing item against a wall, on the floor – using physics to close the loop so sensing is not required.

Closed-loop or open-loop?
How to model uncertainty itself?

Non-deterministic uncertainty:
Keeping track of where item could be, return a worst case set of poses

Probabilistic uncertainty:
Maximize a certain reward function

Non-deterministic or probabilistic uncertainty?
Small scale / unimodal. Often comes with simplifications.

- Complicated models can't be well represented in full state.

Closed-form: Kalman Filter (representing state as a Gaussian distribution)
  - Small scale / unimodal. Often comes with simplifications.
  - Complicated models can't be well represented in full state.

Sample-based representation: Particle filter
  - Large scale / multimodal
  - Comes with added noise

**Closed-form** or **sample-based** representation?
Examples:

**Estimate:** use Kalman filter, take the mean of position estimates and feed it to a deterministic planner

**Reactive:** PID controller, plans only the next step

**Full planning:** plans an optimal path from current state to goal state. (slower)
Type of sensing feedback:
Kinect/camera/depth sensors

Tactile sensing (measure forces being imparted on manipulators)

No Feedback: Using knowledge of environment/physics to reduce uncertainty
Mapping visual error to joint torques

Camera tracks red marker on the wire cutter, and green point on the wire, and attempts to align the coordinates of both the markers.

Robot adjusts joint velocities to reduce the distance between the two points.

Downside: difficult to specify a goal, hard to track wire and wire cutter, need bright colors.
Using depth image from Kinect: same idea, but using 3D depth tracking of arm and door-knob rather than raw RGB images.
realtime tracking of articulated objects

dark grey: depth measurements colored: tracked model

Visual Feedback

Tactile Feedback

No Feedback
Visual Feedback

Tactile Feedback

No Feedback
• Putting key in lock requires millimeter level of uncertainty

• Measure when the key hits the knob, then we only need to move down in one dimension

Use “guarded moves” to reduce uncertainty
Q: How to generate a sequence of such behaviors without first measuring the distances?

A: Generate a set of all possible actions, then generate a sequence of actions that maximizes information gain.

When finger makes contact, we know that the finger must be touching the door handle at a certain position. Try different actions to eliminate subsets of actions that are physically impossible. Gather information first, then execute.
Instead of sampling every point in the image, sample only regions where there is contact with an object.

Estimate the pose of the object using tactile sensing


Estimate the configuration of the robot using tactile sensing

Using force profiles to servo planned motion.

**Closed-loop grasping using contact sensing**


Goal of the learned movement
Misplaced object

Learn feedback policies that use sensor feedback

Learn a policy to generate motor commands to fit a target image from the camera.

Learn feedback policies that use sensor feedback

Tactile Feedback:
Model based approaches -> generalize well, but performance capped by model accuracy
Learning based approach -> doesn’t generalize well but achieves good performance for the specific task.
Tray tilting
- Push strategy
- Model filter
- Probabilistic trajectory selection
- Convergence planning

No feedback –
- Use knowledge of physics
  - Tray tilting
  - Push strategy
  - Model filter
  - Probabilistic trajectory selection
  - Convergence planning

Visual Feedback

Tactile Feedback

No Feedback
An example of non-deterministic planning

Tilting a tray to align objects based on how it will react to physics -> it can be proved that it takes at most 9 moves to get to a certain state

Open-loop robotic part alignment

Use single DOF motions and while expecting objects to roll into the grasp.

**Push Grasping**

Push objects around to reduce the uncertainty about their positions until a path to the goal object can be planned.

Rearrangement Planning

Sample multiple trajectories in simulation and evaluate their probability of success.
Convergent Planning: Compute divergence for each path based on dynamics of objects
A brief introduction to POMDPs.

- Quasi-static model: most of actions in human-scale are slow enough that dynamics can be ignored.
- Using Actions & Observations: What is \( P(S) \), the probability of being in a certain state?
- Given original state \( S_0 \) and final state \( S_f \), generate optimal sequence of actions \( A_{opt} \) for to maximize reward \( R \)
- Plan in belief space, no state space
  - Offline planning: Point-based policy optimization
  - Online planning: Sample-based policy making ("action tree" decision/observation)
State: joint positions, velocities
Action: commanded joint accelerations/torques
Observations: joint position sensors, camera images etc.

\[
\begin{align*}
\text{state} & \quad s = (q, x) \\
\text{action} & \quad a = (\dot{q}, \Delta t) \\
& \quad T = p(s'|s, a) \\
\text{observation} & \quad o = (o_q, o_c) \\
& \quad \Omega = p(o|s, a)
\end{align*}
\]
state space: \[ \dim(S) = n \]

Planning in Belief Space
Belief space can represent any noisy, stochastic position.
Offline Planning

Point-Based Methods

Online Planning
Offline Planning:

Policies are pre-computed
Given a belief and a policy, evaluate value function for a sequence of actions and states.

Very slow to compute offline but efficient to use online.

Point-based solvers

\[ V^\pi = \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t) \]

Point-based solvers


\[ V^\pi = \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t) \]

Point-based solvers

\[ V^\pi = \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t) \]

Point-based solvers

\[ V^\pi = \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t) \]

Point-based solvers

\[ V^\pi = \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t) \]

**Point-based solvers**


\[ \pi^* = \arg \max_{\pi} V^\pi [b(s_0)] \]

Point-based solvers

Offline Planning

Point-Based Methods

Online Planning

\[ V^\pi \rightarrow V^* \]
Online planning: Sample-based policy making ("action tree" decision/observation)
\[ a \sim \pi_{\text{explore}}(b_0) \]
\[ s' \sim T(s, a, s') \]
\[ o \sim p(o|s, a) \]

Offline Planning
*Point-Based Methods*

Online Planning
The post-contact belief space is small

The post-contact belief space is small
The post-contact belief space is small

The post-contact belief space is small

Decompose into pre- and post-contact policies

Belief 3

Offline Planning

Point-Based Methods

Online Planning
One approach in combining the two is to use offline planning policies as heuristics to inform online search.

Heuristics / Bounds

Combine online and offline planning.
Physics-Based Grasping under Uncertainty

Michael Koval
mkoval@cs.cmu.edu

February 1, 2017
16-662 Robot Autonomy