Reinforcement Learning on Variable Impedance Controller for High-Precision Robotic Assembly

Presenter: Weihao Zeng

April 14, 2020
Outline

• Background Info
  • Introduction
  • Related work
• Method
  • High Level
  • Detail
• Experiments
  • Performance
  • Generalization
• Conclusion
Back Info – Introduction

- Current robotics control methods in industry
  - Low precision
  - Does not generalize
- Goal of this paper
  - Use operational space control
  - Incorporate impedance (Force, Torque) in control
  - Leverage NN for generalization
Back Info – Related Works

• Guided policy search (GPS)
  • Not suitable for high precision task since cannot avoid local optima
• LSTM learn two separate policy for finding and inserting peg
  • Require pre-defined heuristics
  • Action space is discrete
• Combine RL with motion planner
  • Learn trajectory following torque controller
  • Assumes access to trajectory planner that avoid local optima
  • Encode planned references into NN with attention mechanism
• Good generalization in simulation
Outline

- Background Info
  - Introduction
  - Related work
- Method
  - High Level
  - Detail
- Experiments
  - Performance
  - Generalization
- Conclusion
Method – High Level

• One sentence summary
  • Approximate an existing control algorithm with NN taking the form $\pi(x_t, F/T_t) \rightarrow u_t$ with a skip connection for Force/ Torque.
• Using iLQG(iterative linear quadratic Gaussian)
• Skip connecting force torque information
Method – Detailed (Outline)

• Setup
• Solution
  • Operational space Controller
  • iLQG
  • Interpret as constraint proposal
  • NN with MDGPS
Method – Setup

• Find a trajectory that minimizes the total cost
  \[
  \min_{u_1 \ldots u_{T-1}} \sum_{t=1}^{T} l(x_t, u_t)
  \]
  \[
  x_{t+1} = f(x_t, u_t), t = 1 \ldots T - 1
  \]

• Operational Space
  • In this setting, cartesian space for the manipulator
  • \( F_{tip} = [F_x, F_y, F_z, M_x, M_y, M_z] \) (wrench for end-effector)
  • The requested force
Method – Op. Space Controller

\[ M(q)\ddot{q} + c(q, \dot{q})\dot{q} + g(q) + J^T(q)F_{tip} = \tau, \quad (1) \]

- Only consider gravity and gripper contact force for external force
Method – Op. Space Controller

\[ M(q)\ddot{q} + c(q, \dot{q})\dot{q} + g(q) + J^T(q)F_{tip} = \tau, \]  
\[ g(q) + J^T(q)F_{tip} = \tau, \]
\[ \tau = g(q) + J^T(q)F_{tip} + [I - J^T(q)J^{T\dagger}(q)]\tau_{null}, \]

- Only considering gravity and gripper contact force for external forces.
- Since the robots will move rather slowly during manipulation, we will ignore the higher order terms (acceleration, velocity)
- Project the torque to its non-empty null space.
Method – Op. Space Controller

\[ \tau = g(q) + J^T(q)F_{tip} + [I - J^T(q)J^T(q)]\tau_{null}, \] (2)

- **Remark**
  - Assuming no contact other than tip
  - Robots move slow
- **Problem**
  - Result in continuous acceleration in application
Method – Op. Space Controller

\[ \tau = \Sigma_1 [K_{qp}(q - q^*) + K_{qd}(\dot{q} - \dot{q}^*)] + \Sigma_2 J^T(q)F_{tip} \]  
\[ + [I - J^T(q)J^{T\dagger}(q)]\tau_{null} + g(q) , \]  

- PD Control
  - \(K_{qp}, K_{qd}\) are diagonal gain matrices.
  - \(q^*, \dot{q}^*\) are desired joint positions and velocity
- Weighing motion and force factors
  - \(\Sigma_1, \Sigma_2\) weights motion and force factors respectively
Method – iLQG

• The existing controller we are approximating.
• A MPC (model predictive control)
• Setup
  • Input:
    • Cost function $J(w) = \sum_{t=1}^{T} J(x_t, u_t)$, Dynamics $f(x_t, u_t) = x_{t+1}$
    • Trajectory $w = \{(x_1, u_1), \ldots, (x_T, u_T)\}$
  • Output
    • A new optimized trajectory $w'$ that minimizes $J(w')$
    • Take first action($u_1$) in this trajectory.
One sentence summary
• Given a trajectory \{ (x_1, u_1), \ldots (x_T, u_T) \}, update the trajectory backward with dynamic programming (backward pass); each update takes local second order linear approximation and update (Gauss Newton’s method); calculate a new trajectory from initial state and new actions; and repeat.
Method – iLQG (glimpse of math)

\[
J_i(x, U_i) = \sum_{j=i}^{N-1} \ell(x_j, u_j) + \ell_f(x_N)
\]

Two types of cost function:
- Running (non-final state)
- Final (only final state)

\[
V(x, i) = \min_u [\ell(x, u) + V(f(x, u), i+1)]
\]

Dynamic programming, update one step a time.

\[
Q(\delta x, \delta u) = \ell(x + \delta x, u + \delta u, i) - \ell(x, u, i) + V(f(x + \delta x, u + \delta u), i+1) - V(f(x, u), i+1)
\]

Approximation around given \(x, u\) with small perturbations of \(x, u\).

Second order Taylor expansion:

\[
\approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}
\]
Method – iLQG (glimpse of math)

\[ Q_x = \ell_x + f_x^T V'_x \]  
\[ Q_u = \ell_u + f_u^T V'_x \]  
\[ Q_{xx} = \ell_{xx} + f_x^T V'_{xx} f_x + V'_x \cdot f_{xx} \]  
\[ Q_{uu} = \ell_{uu} + f_u^T V'_{uu} f_u + V'_x \cdot f_{uu} \]  
\[ Q_{ux} = \ell_{ux} + f_u^T V'_{xx} f_x + V'_x \cdot f_{ux} \cdot \]  

\[ \Delta V(i) = -\frac{1}{2} Q_u Q_{uu}^{-1} Q_u \]  
\[ V_x(i) = Q_x - Q_u Q_{uu}^{-1} Q_{ux} \]  
\[ V_{xx}(i) = Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{ux} \cdot \]  

\[ k = -Q_{uu}^{-1} Q_u \]  
\[ K = -Q_{uu}^{-1} Q_{ux} \]  

\[ \hat{x}(1) = x(1) \]  
\[ \hat{u}(i) = u(i) + k(i) + K(i) (\hat{x}(i) - x(i)) \]  
\[ \hat{x}(i+1) = f(\hat{x}(i), \hat{u}(i)) \]
Method – iLQG

• Each update is updating a Gaussian normal
  \[ p(u_t|x_t) = N(K_t x_t + k_t, C_t), C_t = Q_{u,u}^{-1} \]
  \[ k = -Q_{uu}^{-1}Q_u \]
  \[ K = -Q_{uu}^{-1}Q_{ux} \]

• iLQG Application in this paper
  • Added entropy so more likely to explore
  \[ \tilde{l}(x_t, u_t) = l(x_t, u_t) - H(p(u_t|x_t)) \]
  • State(x) varies between experiments but generally joint position and velocity.

• Summary
  • Using iLQG (the existing controller) for control.
  • Backward pass – altering normal distributions
  • Forward pass – calculate new trajectories
Method – implicit constraint proposal

\[ M(q)\ddot{q} + c(q, \dot{q})\dot{q} + g(q) + J^T(q)F_{tip} = \tau, \quad (1) \]

• Rewrite operation space control dynamics (wrench) in motion (twist)
  \[ F = \Lambda(q)\dot{V} + \eta(q, V) \]
  • \( V \in SE(3), V \in R^6 \)
• Kinematics term
  • \( \Lambda(q) = J^{-T}(q)M(q)J^{-1}(q) \)
• Coriolis term and others
  • \( \eta(q, V) = J^{-T}(q)c(q, J^{-1}V) - \Lambda(q)J(q)J^{-1}(q) \)
Method – implicit constraint proposal

\[ \mathcal{F} = \Lambda(q) \dot{V} + \eta(q, V) + A^T(q) \lambda, \]

- Add Pfaffian constraints
  - \( A(q)V = 0 \)
- Need new \( A(q) \) for every new task – bad, bad
  - View \( A(q) \) as description for the task
    - E.g. pushing peg
  - Use NN to (implicitly) learn \( A(q) \) through interactions
Method – NN with MDGPS

$$\min \mathbb{D}_{KL}(\pi_\theta(u_t|x_t) \| p(u_t|x_t)) \quad \forall x_t, u_t, t,$$

- Supervised learning for training (MDGPS)
  - Mirror descent guided policy search
- Skip connection F/T(force torque) into last hidden layer
  - Force torque should not alter NN’s interpretation for robot state
Algorithm 1 Force-based RL controllers

1: for iteration $k \in \{1, \ldots, K\}$ do
2:   Train local RL controller using iLQG, where $u_t$ is set as operational space force controller
3:   Project calculated operational control to joint torque using Eq.3
4:   Train neural network controller using MDGSP[15]
5: end for
Method – Summary

• Setup
  • Minimize cost over trajectory

• Solution
  • Operational space controller
    • Mix motion (PD control) and force; ignore motion in dynamics
  • iLQG
    • Iterative backward Gauss Newton update
  • Interpret as constraint proposal
    • NN learns constraints for tasks
  • NN with MDGPS
    • Skip connection of force, torque to the last hidden layer
Outline

• Background Info
  • Introduction
  • Related work
• Method
  • High Level
  • Detail
• Experiments
  • Performance
  • Generalization
• Conclusion
Experiments – Performance (setup)

- Siemens Robot Learning Challenge gear assembly
- Assume all components are grasped when starting
Experiments – Performance (setup)

- 4 individual tasks
- Independently trained

a) Round peg in round hole.  
b) Gear wheel on shaft.

c) Squared hole on squared shaft.  
d) Teeth Alignment.
Experiments – Video
## Experiments – Performance

<table>
<thead>
<tr>
<th>Method</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinematics with way-points for task 1,2</td>
<td>0/5</td>
<td>0/5</td>
<td>0/5</td>
<td>0/5</td>
</tr>
<tr>
<td>iLQG with torque control</td>
<td>1/5</td>
<td>0/5</td>
<td>0/5</td>
<td>0/5</td>
</tr>
<tr>
<td>iLQG with torque control, state include torque/force</td>
<td>0/5</td>
<td>0/5</td>
<td>0/5</td>
<td>0/5</td>
</tr>
<tr>
<td><strong>Paper’s method</strong></td>
<td>5/5</td>
<td>5/5</td>
<td>2/5</td>
<td>4/5</td>
</tr>
<tr>
<td>Paper’s method, state includes torque/force</td>
<td>5/5</td>
<td>5/5</td>
<td>3/5</td>
<td>3/5</td>
</tr>
</tbody>
</table>
## Experiments – Generalization

<table>
<thead>
<tr>
<th></th>
<th>1 cm</th>
<th>2 cm</th>
<th>5 cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>iLQG</td>
<td>8/10</td>
<td>5/10</td>
<td>6/10</td>
</tr>
<tr>
<td>Paper’s method</td>
<td>9/10</td>
<td>8/10</td>
<td>6/10</td>
</tr>
<tr>
<td>Paper’s method, state includes torque/force</td>
<td>0/10</td>
<td>0/10</td>
<td>0/10</td>
</tr>
</tbody>
</table>

- Only consider task 2
- Good generalization capability relatively
- T/F skip connection is good
Outline

- Background Info
  - Introduction
  - Related work
- Method
  - High Level
  - Detail
- Experiments
  - Performance
  - Generalization
- Conclusion
Conclusion

• Accomplishments
  • RL combined with operational space force controller can solve high precision robotic assembly
  • NN architecture explicitly considering torque and force are good.

• Future work
  • Add raw vision and tactile inputs
  • Experiment with different starting point
  • Model contact explicitly and encode more structured Pfaffian constraint matrix
Thank you :)