UC San Diego

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

$$M(q)\ddot{q} + c(q,\dot{q})\dot{q} + g(q) + J^T(q)\mathcal{F}_{tip} = \tau, \qquad (1)$$

- Only consider gravity and gripper contact force for external force

Method – Op. Space Controller

$$M(q)\ddot{q} + c(q,\dot{q})\dot{q} + g(q) + J^T(q)\mathcal{F}_{tip} = \tau, \qquad (1)$$

$$g(q) + J^T(q)\mathcal{F}_{tip} = \tau,$$

$$\tau = g(q) + J^T(q)\mathcal{F}_{tip} + [I - J^T(q)J^{T\dagger}(q)]\tau_{null}, \qquad (2)$$

- 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

$$\tau = g(q) + J^T(q)\mathcal{F}_{tip} + [I - J^T(q)J^{T\dagger}(q)]\tau_{null}, \qquad (2)$$

- Remark
  - Assuming no contact other than tip
  - Robots move slow
- Problem
  - Result in continuous acceleration in application

Method – Op. Space Controller

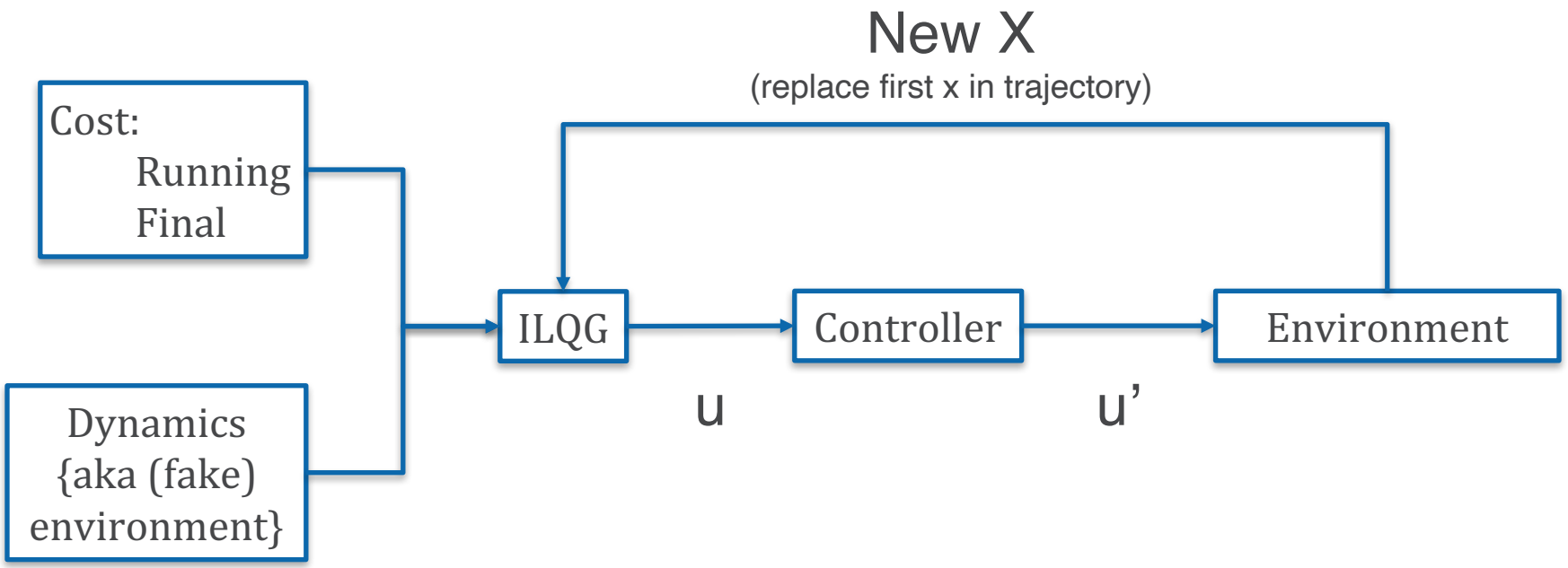
$$\tau = \Sigma_1[K_{qp}(q - q^*) + K_{qd}(\dot{q} - \dot{q}^*)] + \Sigma_2 J^T(q)\mathcal{F}_{tip} \quad (3)$$
$$+ [I - J^T(q)J^{T\dagger}(q)]\tau_{null} + g(q), \quad (4)$$

- PD Control
  - K_{qp}, K_{qd} are diagonal gain matrices.
  - $q^*, \dot{q}^*$ are desired joint positions and velocity
- Weighing motion and force factors
  - $\sum_1, \sum_2$ 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), …(x_T, u_T)}
  - Output
    - A new optimized trajectory w' that minimizes J(w')
  - Take first action(u_1) in this trajectory.

Method – iLQG



One sentence summary

- Given a trajectory {(x_1, u_1), …(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)

$$J_i(\mathbf{x}, \mathbf{U}_i) = \sum_{j=i}^{N-1} \ell(\mathbf{x}_j, \mathbf{u}_j) + \ell_f(\mathbf{x}_N)$$

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

$$V(\mathbf{x}, i) = \min_{\mathbf{u}} [\ell(\mathbf{x}, \mathbf{u}) + V(\mathbf{f}(\mathbf{x}, \mathbf{u}), i+1)]$$

Dynamic programming, update one step a time.

$$\begin{aligned} Q(\delta\mathbf{x}, \delta\mathbf{u}) &= \ell(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}, i) - \ell(\mathbf{x}, \mathbf{u}, i) \\ &\quad + V(\mathbf{f}(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}), i+1) - V(\mathbf{f}(\mathbf{x}, \mathbf{u}), i+1) \end{aligned}$$

Approximation around given x, u with small perturbations of x, u.

$$\approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 0 & Q_{\mathbf{x}}^{\mathsf{T}} & Q_{\mathbf{u}}^{\mathsf{T}} \\ Q_{\mathbf{x}} & Q_{\mathbf{xx}} & Q_{\mathbf{xu}} \\ Q_{\mathbf{u}} & Q_{\mathbf{ux}} & Q_{\mathbf{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}$$

Second order Taylor expansion.

Method – iLQG (glimpse of math)

$$
\begin{aligned}
Q_{\mathbf{x}} &= \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} V_{\mathbf{x}}' && \text{(5a)}\\
Q_{\mathbf{u}} &= \ell_{\mathbf{u}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V_{\mathbf{x}}' && \text{(5b)}\\
Q_{\mathbf{xx}} &= \ell_{\mathbf{xx}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} V_{\mathbf{xx}}' \mathbf{f}_{\mathbf{x}} + V_{\mathbf{x}}' \cdot \mathbf{f}_{\mathbf{xx}} && \text{(5c)}\\
Q_{\mathbf{uu}} &= \ell_{\mathbf{uu}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V_{\mathbf{xx}}' \mathbf{f}_{\mathbf{u}} + V_{\mathbf{x}}' \cdot \mathbf{f}_{\mathbf{uu}} && \text{(5d)}\\
Q_{\mathbf{ux}} &= \ell_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V_{\mathbf{xx}}' \mathbf{f}_{\mathbf{x}} + V_{\mathbf{x}}' \cdot \mathbf{f}_{\mathbf{ux}}. && \text{(5e)}
\end{aligned}
$$

Calculations
Backward pass

$$
\begin{aligned}
\Delta V(i) &= -\tfrac{1}{2} Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} && \text{(7a)}\\
V_{\mathbf{x}}(i) &= Q_{\mathbf{x}} - Q_{\mathbf{u}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} && \text{(7b)}\\
V_{\mathbf{xx}}(i) &= Q_{\mathbf{xx}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}. && \text{(7c)}
\end{aligned}
$$

$$
\mathbf{k} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}}
$$

$$
\mathbf{K} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}
$$

$$
\begin{aligned}
\hat{\mathbf{x}}(1) &= \mathbf{x}(1)\\
\hat{\mathbf{u}}(i) &= \mathbf{u}(i) + \mathbf{k}(i) + \mathbf{K}(i)(\hat{\mathbf{x}}(i) - \mathbf{x}(i))\\
\hat{\mathbf{x}}(i+1) &= \mathbf{f}(\hat{\mathbf{x}}(i), \hat{\mathbf{u}}(i))
\end{aligned}
$$

Update
Forward Pass

Method – iLQG

- Each update is updating a Gaussian normal
  - $p(u_t|x_t) = N(\boldsymbol{K}_t x_t + k_t, \boldsymbol{C}_t), \boldsymbol{C}_t = Q_{u,u}^{-1}$

$$\mathbf{k} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}}$$

$$\mathbf{K} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}$$

- 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 trajactories

Method – implicit constraint proposal

$$M(q)\ddot{q} + c(q,\dot{q})\dot{q} + g(q) + J^T(q)\mathcal{F}_{tip} = \tau, \qquad (1)$$

- 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
    - $\eta(q,V) = J^{-T}(q)c(q,J^{-1}V) - \Lambda(q)J(\dot{q})J^{-1}(q)$

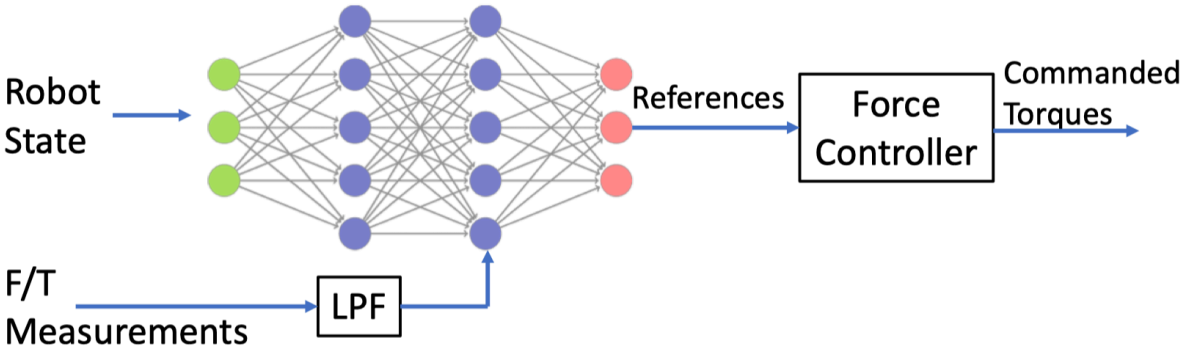
Method – implicit constraint proposal

$$\mathcal{F} = \Lambda(q)\dot{\mathcal{V}} + \eta(q, \mathcal{V}) + \underbrace{A^T(q)\lambda}_{\mathcal{F}_{tip}}, \qquad (7)$$

- 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

$$\min \mathbb{D}_{KL}(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)||p(\mathbf{u}_t|\mathbf{x}_t)) \quad \forall \mathbf{x}_t, \mathbf{u}_t, t,$$



- 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

Method – NN with MDGPS

Algorithm 1 Force-based RL controllers

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

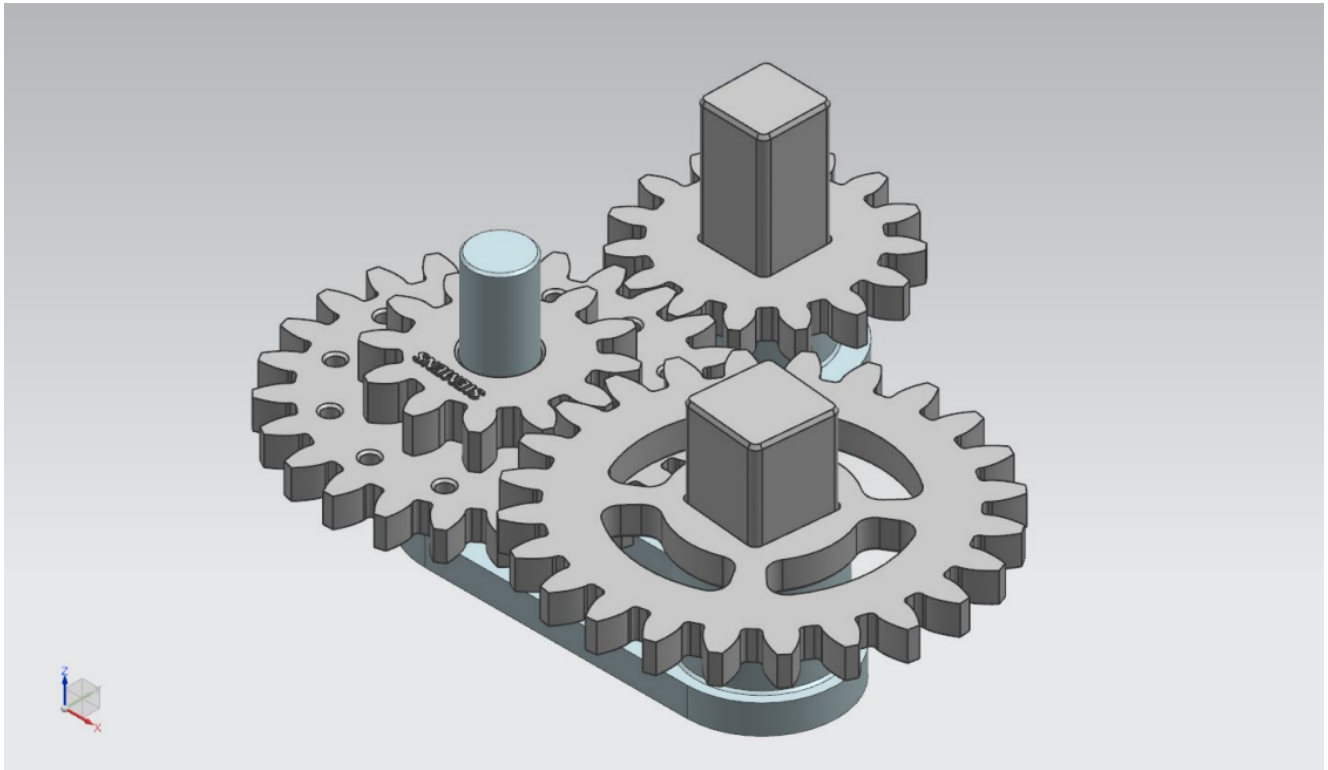
Method – Summary

- 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

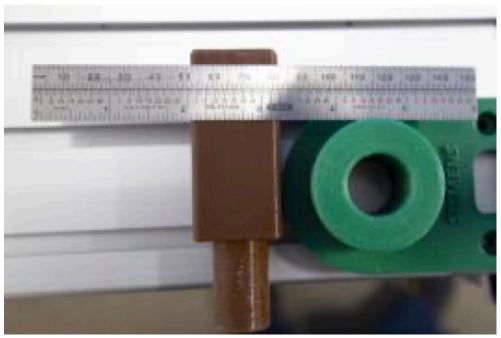
- Background Info
  - Introduction
  - Related work
- Method
  - High Level
  - Detail
- **Experiments**
  - Performance
  - Generalization
- Conclusion

Experiments – Performance (setup)



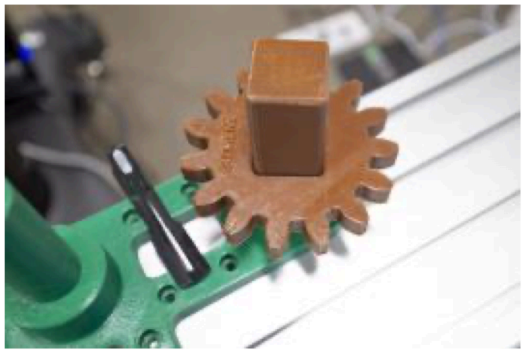
- Siemens Robot Learning Challenge gear assembly
- Assume all components are grasped when starting

Experiments – Performance (setup)

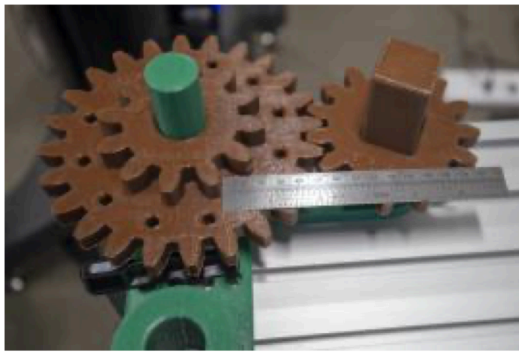


a) Round peg in round hole.

b) Gear wheel on shaft.



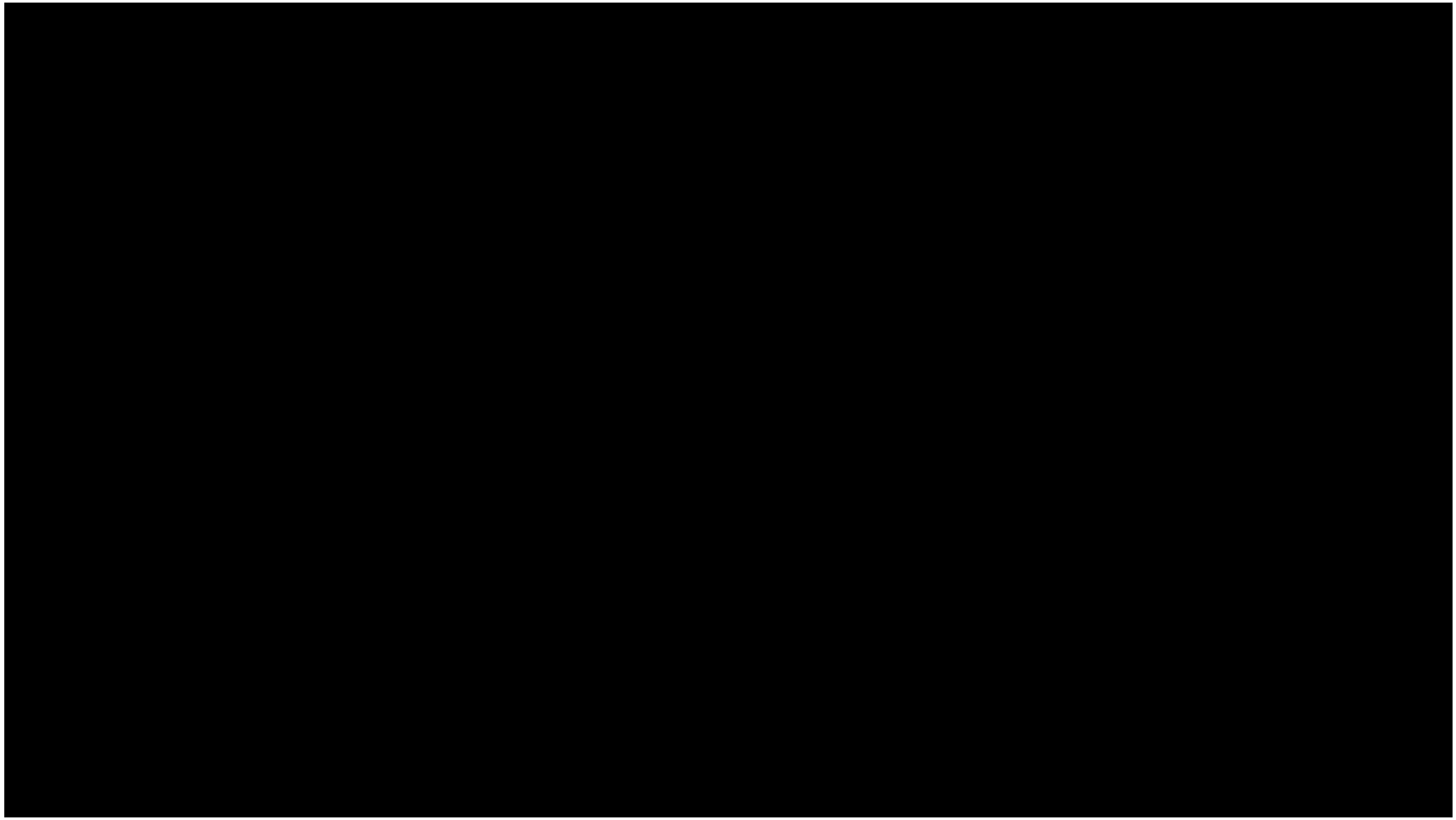
c) Squared hole on squared shaft.



d) Teeth Alignment.

- 4 individual tasks
- Independently trained

Experiments – Video



Experiments – Performance

	Task 1	Task 2	Task 3	Task 4
Kinematics with way-points for task 1,2	0/5	0/5	0/5	0/5
iLQG with torque control	1/5	0/5	0/5	0/5
iLQG with torque control, state include torque/force	0/5	0/5	0/5	0/5
Paper's method	**5/5**	**5/5**	2/5	**4/5**
Paper's method, state includes torque/force	5/5	5/5	**3/5**	3/5

Experiments – Generalization

	1 cm	2 cm	5 cm
iLQG	8/10	5/10	**6/10**
Paper's method	**9/10**	**8/10**	**6/10**
Paper's method, state includes torque/force	0/10	0/10	0/10

- 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 staring point
  - Model contact explicitly and encode more structured Pfaffian constraint matrix

Thank you :)