A Framework for Robot Manipulation: Skill Formalism, Meta Learning and Adaptive Control

Presenter: Riley Hadden

5/19/20

Outline

- Introduction
- Related work
- Method
- Experiments
- Conclusion

Introduction

Motivation

- Humans move with the contraction of muscle fibers.
  - Average adult male bicep has 250,000 such fibers
- We do not consciously control individual fibers
- Learn motor programs which aggregate signals sent to fibers e.g. reach, grasp, lift
- Aggregate motor programs into more complicated programs e.g. sip coffee from cup on table
- Parameterize motor programs with current goals such as grasp glass in given pose relative to me

Introduction

Framework using human motor control as a model:

1. Adaptive Impedance Controller
    - Utilize localized intelligence

2. Define formal representation of manipulation skill
    - Breaking down skill into manipulation primitives
    - Links desired actions to adaptive impedance control
    - Reduce complexity of task

3. Apply learning techniques to tune
    - Solve for meta parameters instead of motor control

Related Work

	Key Insight	Short Coming	Extension
Manipulation Skill from motion primitives	Break down complex task into movement primitives	Arduous manual tuning phase	Formalizing skills and manipulation primitives allows for learning algorithm to tune
RL	Apply RL to learn parameters for completing manipulation skills	Computationally expensive, relying on GPUs	High level skill formalization coupled with low level impedance controller reduce complexity
Impedance controller	Localized intelligence for movement	General form could not be applied here	Extended to Cartesian space and full feed-forward tracking

Method: Overview

Framework consists of three core components:

1. Controller for Motion Primitives
   - Adaptive Impedance Controller chosen here

2. Manipulation Skill Formalism
   - Break complex problem into series of motion primitives

3. Learning Algorithm for tuning
   - Learn hyper parameters not complex control

Method: Adaptive Impedance Controller

Rigid Body Dynamics:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau_u + \tau_{ext}$$

$M(q)$ mass matrix

$C(q, \dot{q})$ Coriolis/Centrifugal torques

$g(q)$ gravity

τ_{ext} vector of external link-side joint torques

Adaptive impedance control law:

$$\tau_u = J(q)^T(-F_{ff}(t) - F_d(t) - K(t)e - D\dot{e}) + \tau_r$$

$F_{ff}(t)$ - adaptive feed forward wrench

$F_d(t)$ - optional feed forward wrench trajectory

$K(t)$ - stiffness matrix

$J(q)$ - Jacobian

$e = x^* - x$ - (position error)

$\dot{e} = \dot{x}^* - \dot{x}$ - (velocity error)

τ_r - dynamics compensator

Method: Adaptive Impedance Controller

Adaptive impedance control law:

$$\tau_{ext} = J(q)^T(-F_{ff}(t) - F_d(t) - K(t)e - D\dot{e}) + \tau_r$$

Adaptive tracking error:

$$\epsilon = e + \kappa \dot{e}$$

Adaptive feed forward wrench:

$$F_{ff} = \int_0^t \dot{F}_{ff}(t)dt + F_{ff}(0)$$

Adaptive stiffness:

$$K(t) = \int_0^t \dot{K}(t)dt + K(0)$$

$$\dot{F}_{ff}(t) = \frac{1}{T}\alpha(\epsilon - \gamma_a(t)F_{ff}(t)), \quad \dot{K}(t) = \frac{1}{T}\beta(diag(\epsilon \circ \epsilon) - \gamma_\beta(t)K(t))$$

α, β - Learning rates for feed forward wrench and stiffness respectively (adaptation speed)

$\gamma_\alpha, \gamma_\beta$ - Forgetting factors for feed forward wrench and stiffness respectively (slow down adaptation process)

D - Cartesian Dampening from:

A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger, “Cartesian ¨ impedance control of redundant robots: Recent results with the DLRlight-weight-arms,”

T - Sample time of controller

Method: Adaptive Impedance Controller

Adaptive impedance control law:

$$\tau_u = J(q)^T(-F_{ff}(t) - F_d(t) - K(t)e - D\dot{e}) + \tau_r$$

Policy:

$$\tau_u = f(\dot{x}_d, F_d, \alpha, \beta, \gamma_\alpha, \gamma_\beta, \Omega)$$

Ω - precept vector (e.g. pose, velocity, forces, etc.)

Method: Adaptive Impedance Controller

Policy:

$$\tau_u = f(\dot{x}_d, F_d, \alpha, \beta, \gamma_\alpha, \gamma_\beta, \Omega)$$

Adaptive tracking error:

$$\epsilon = e + \kappa\dot{e}$$

Constraining Parameters:

$$\dot{K}_{max} = max_{t>0}\frac{1}{T}[\beta(\epsilon(t)\epsilon(t) - \gamma_b K(t))]$$

Assume: $K(0) = 0$ and $\dot{e} = 0$

e_{max} := error at which $\dot{K}_{max} = \dfrac{\beta e_{max}^2}{T}$

$$\beta_{max} := \frac{\dot{K}_{max}T}{e_{max}^2}$$

Maximum decrease of stiffness occurs when $e = 0$ and $K(t) = K_{max}$

$$\gamma_{\beta,max} := \frac{\dot{K}_{max}}{\beta_{max}K_{max}}$$

Method: Manipulation Skill Formalism

Overview

1. Manipulation Skill (high level out come to achieve)
    - Directed graph consisting of Manipulation Primitives

2. Conditions
    - determine transitions between nodes

3. Learning Metric
    - result, $r = \{0,1\}$ 0 - failure / 1 - success
    - q cost function of the skill
        - e.g. for peg in hole, this could be insertion time or average contact forces

Method: Manipulation Skill Formalism

A *manipulation skill* is a directed graph G consists of

- nodes $n \in N$, *manipulation primitives* (MP)
- edges $e \in E$, transitions

MPs consist of:

- $\dot{x}_d = f_t(P_t, \Omega)$ (parameterized twist)
- $F_d = f_w(P_w, \Omega)$ (feed forward wrench trajectory)
- (both with respect to Task Frame TF)

Ω - precept vector (e.g. current pose, external forces,…)
P_t - set of all parameters used by node n to generate twist
P_w- set of all parameters used by node n to generate wrench

Method: Manipulation Skill Formalism

Robot Manipulation Skill Formalism

Moving through the graph:

1. Execute current node n_i
2. Evaluate current condition
3. Select transition (edge) to $n_{i+1}$by condition
4. Repeat until terminal condition reached

Conditions:

1. C_{pre} - Precondition : conditions under which skill can be initialized (success may be reached from here)
2. C_{err} - Error Condition: stops execution and returns a negative result
3. C_{suc} - Success Condition: conditions under which a positive result has been achieved

Method: Manipulation Skill Formalism

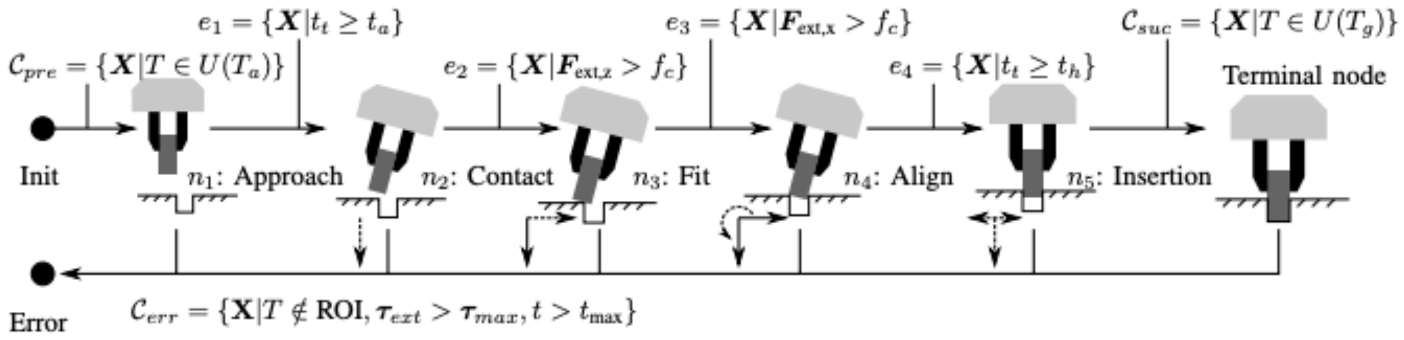

Fig. 3: Visualization of the peg-in-hole skill graph. Dashed arrows denote velocity commands, solid ones feed forward force commands. t_1 and t_4 are the trajectory durations in states n_1 and n_4.

Peg in Hole Example

n_1: Move to canonical approach pose (above hole)

$$\dot{x}_d = f_{p2p}(T_a, s\dot{x}_{max}, \ddot{x}_{max}), F_d = 0$$

f_{p2p} determined by impedance controller

n_2: Moves toward surface with hole

$$\dot{x}_d = [0,0,s\dot{x}_{max},0,0,0]^T, F_d = 0$$

n_3: Object moved laterally in x-direction until constrained

$$\dot{x}_d = [s\dot{x}_{max},0,0,0,0,0]^T F_d = [0,0,f_c,0,0,0]^T$$

n_4: Object is rotated to estimated orientation of hole while maintaining contact

$$\dot{x}_d = f_{p2p}(T_h, s\dot{x}_{max}, \ddot{x}_{max}),$$

$$F_d = [f_c,0,f_c,0,0,0]^T$$

n_5: Object inserted into the hole

$$\dot{x}_d = f_{p2p}(T_s, s\dot{x}_{max}, \ddot{x}_{max}), F_d = 0$$

Method: Parameter Learning

Learning Algorithm constraints:

1. Does not depend on gradient descent
2. Assume stochasticity
3. Global optimizer (no assumptions on convexity of cost function)
4. Ideally would also handle unknown constraints

TABLE II: Suitability of existing learning algorithms with regard to the properties no gradient (NG), stochasticity assumption (SA), global optimizer (GO) and unknown constraints (UC).

Method	NG	SA	GO	UC
Grid Search	+	−	+	−
Pure Random Search	+	−	+	−
Gradient-descent family	−	−	−	−
Evolutionary Algorithms	+	−	+	−
Particle Swarm	+	+	+	−
Bayesian Optimization	+	+	+	+

Method

Parameter Learning

Suitable Algorithms Explored:

1. Bayesian Optimization (BO)
   - Finds minimum of unknown objective function, $f(p)$, by approximating a statistical model of $f(p)$
2. Latin Hypercube Sampling:
   - Random sampling except samples equally distributed in parameter space
3. Covariance Matrix Adaptation and Evolutionary Strategies
   - initial centroid $m \in \mathbb{R}^n$, population size λ, step size $\sigma > 0$ chosen by user
   - initial covariance $C = I$, isotropic and anisotropic paths $p_\sigma = 0$, $p_c = 0$
     a) Evaluate λ individuals sampled from mean m, covariance σC
     b) Updated m, p_σ, p_c, C, and σ by fitness of samples
4. Particle Swarm Optimization
   - initialize particles with positions $x_i(0)$ and velocities $v_i(0)$ drawn from uniform distribution
   - track p_i - personal best and g - global best
     a) update velocity $v_i(t+1) = v_i(t) + c_1(p_i - x_i(t))R_1 + v_i(t) + c_2(g - x_i(t))R_2$
     b) update position $x_i(t+1) = x_i(t) + v_i(t+1)$
     c) evaluate fitness $f(x_i(t+1))$
     d) update p_i g as appropriate

Experiments

Peg in hole:

1. Equilateral triangle <0.1mm tolerance
2. Key
3. Peg << 0.1 mm tolerance

Learning Algorithms:

1. LHS (Latin Hypercube Sampling)
2. CMA-ES (Covariance Matrix Adaptation Evolutionary Strategy)
3. PSO (Particle Swarm Optimization)
4. Bayesian Optimization

Cost:

execution time from first contact to full insertion

Fig. 5: Experimental setup, puzzle (top right), key (bottom left) and peg (bottom right).

Experiments

Results:

- All found feasible solutions for simple tasks
  - Including LHS suggesting complexity reduction was successful
- Bayesian Optimization on average found worse solutions with notably larger confidence intervals
- BO Could not find solution to more complex tasks
- CMA-SE found solutions with lowest absolute cost and smallest confidence interval on average

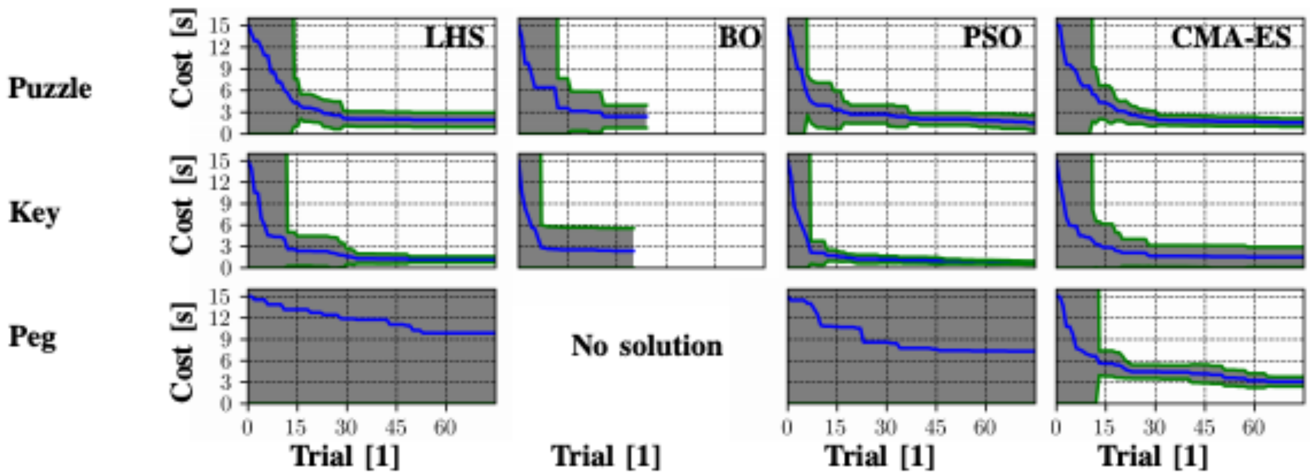

Fig. 6: Experimental results. Columns show the plots for the learning algorithms (LHS,BO,PSO,CMA-ES) and rows for the tasks (Puzzle,Key,Peg).

Conclusion

- Adaptive Impedance Controller and formal description of manipulation skills reduces complexity
- Learning methods and random sampling over reduced solution space feasible
- Fast! With CMA-ES feasible solution in 2-4 minutes, and optimized after 5-20 minutes
- After optimization, performs actual task faster than humans and at industrial precision levels
- Reduced computational demand
  - Less GPU intensive
  - Good for autonomous systems