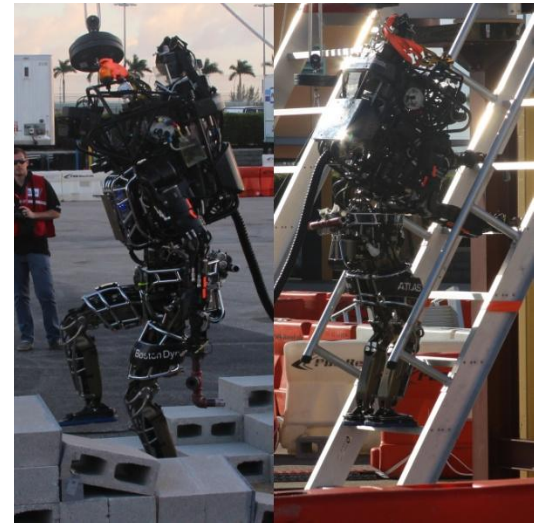
Full Body Control for the Atlas robot

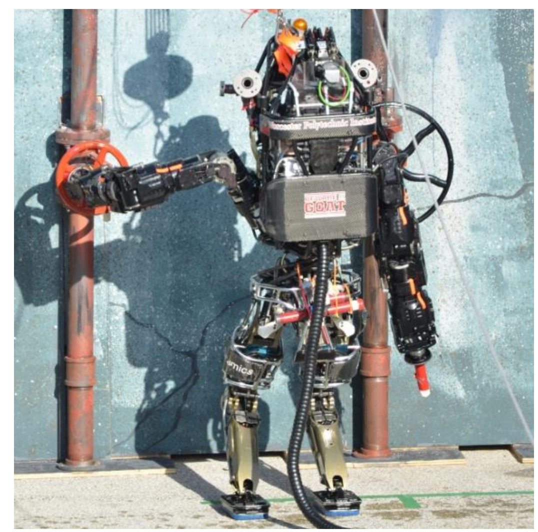
Siyuan Feng
Carnegie Mellon University

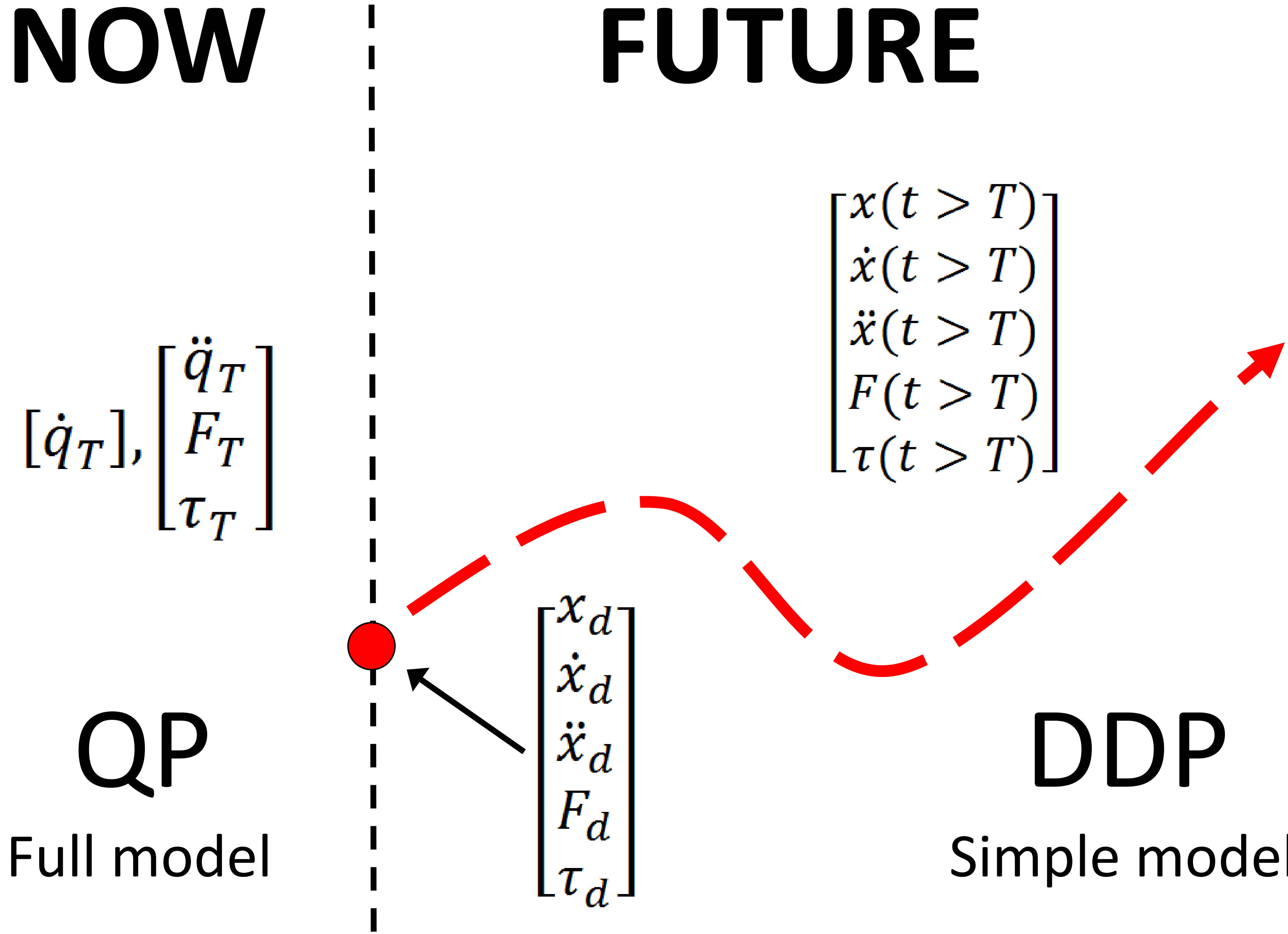
Sep.18 2014

Introduction

- We want a generalized controller that works for a variety of tasks.
- This is a complex optimization problem, so we factor it into two stages.





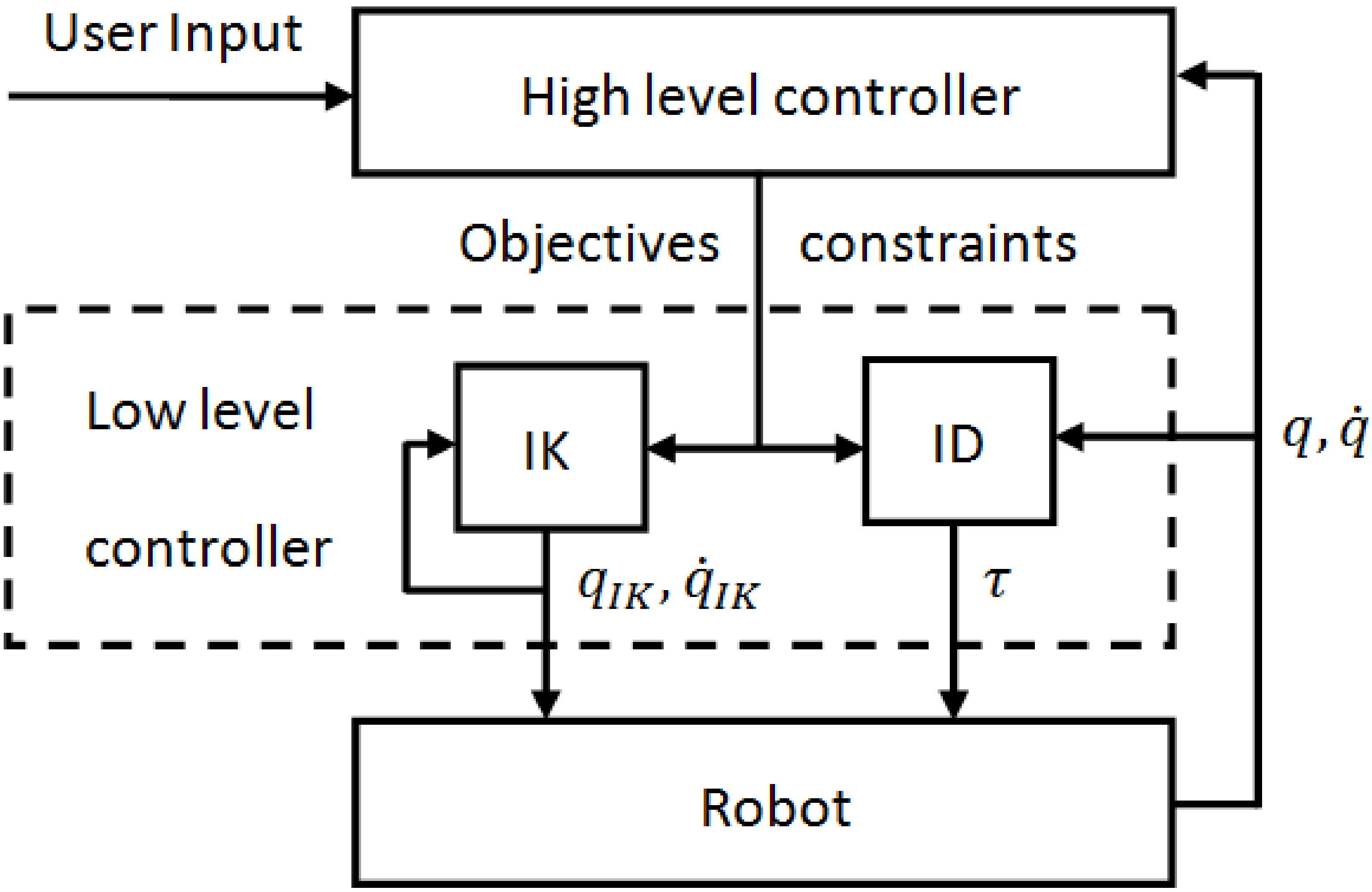

NOW
FUTURE
$[\dot{q}_T], \begin{bmatrix} \ddot{q}_T \\ F_T \\ \tau_T \end{bmatrix}$
$\begin{bmatrix} x(t > T) \\ \dot{x}(t > T) \\ \ddot{x}(t > T) \\ F(t > T) \\ \tau(t > T) \end{bmatrix}$
$\begin{bmatrix} x_d \\ \dot{x}_d \\ \ddot{x}_d \\ F_d \\ \tau_d \end{bmatrix}$
QP
Full model
DDP
Simple model

Related work

- Reference motion generation (Long term)
  - Preview control
  - Instantaneous capture point
  - Centroidal momentum
  - MPC
  - Dynamic programming
- Inverse dynamics (Instantaneous)
  - Operational space control
  - Prioritized inverse dynamics
  - Variable reduction

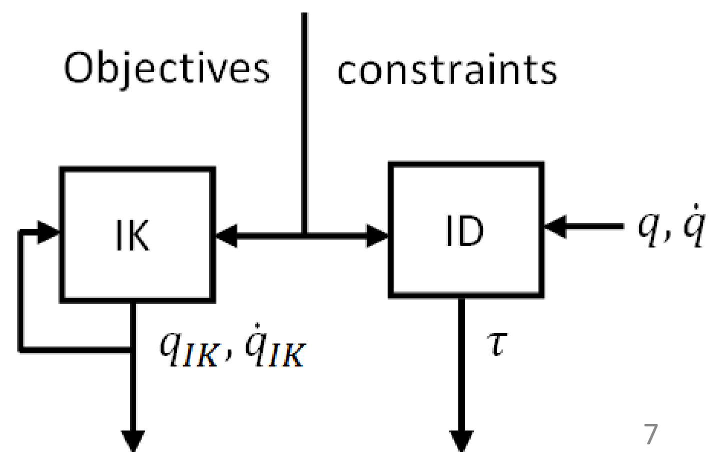
Outline

- Related work
- Full body controller
  - Inverse dynamics
  - Inverse kinematics
  - State estimator
- Task level controllers
  - Walking
  - Ladder climbing
  - Manipulation
- Discussion and work in progress


User Input
High level controller
Objectives
constraints
Low level
controller
IK
ID
$q, \dot{q}$
$q_{IK}, \dot{q}_{IK}$
τ
Robot

Low level controller

- Both inverse dynamics and inverse kinematics
  - ID for compliance
  - IK to battle modeling errors
- ID and IK are formulated as quadratic programming problems optimizing for the current time step.
- Based on the full model (floating base)



QP formulation

$$\min_{\mathcal{X}} \; 0.5\mathcal{X}^T G \mathcal{X} + g^T \mathcal{X}$$
$$s.t. \; C_E \mathcal{X} + c_E = 0$$
$$C_I \mathcal{X} + c_I >= 0$$

Cost function:

$$0.5||A\mathcal{X} - b||^2$$

Each row represents an objective:

$$A = \begin{bmatrix} w_0 A_0 \\ w_1 A_1 \\ \vdots \\ w_n A_n \end{bmatrix}, b = \begin{bmatrix} w_0 b_0 \\ w_1 b_1 \\ \vdots \\ w_n b_n \end{bmatrix}$$

Inverse dynamics

- $\mathcal{X} = [\ddot{q} \quad \tau \quad F]^T$
- Use desired motions from the high level controller and estimated robot states to compute target acceleration $\ddot{x}^*$ with
  $\ddot{x}^* = K_{id}(x_d - x) + D_{id}(\dot{x}_d - \dot{x}) + \ddot{x}_d$
- $\ddot{x} = J\ddot{q} + \dot{J}\dot{q}$

Inverse Dynamics (QP)

Objectives:

- Task objectives
- CoM acceleration
- Change of angular momentum
- Reference pose tracking
- Regularize controls / acc.

$$\begin{bmatrix} w_1 A_1 \\ w_2 A_2 \\ \vdots \\ w_N A_N \end{bmatrix} \begin{bmatrix} \ddot{q} \\ F \\ \tau \end{bmatrix} = \begin{bmatrix} w_1 b_1 \\ w_2 b_2 \\ \vdots \\ w_N b_N \end{bmatrix}$$

$$A_{CoM} = w_{CoM} [J_{CoM} \quad 0 \quad 0]$$

$$b_{CoM} = w_{CoM} (\ddot{x}_{CoM} - \dot{J}_{CoM} \dot{q})$$

Inequality constraints:

- CoP
- Friction cone
- Joint torque

$$\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_M \end{bmatrix} \begin{bmatrix} \ddot{q} \\ F \\ \tau \end{bmatrix} \geq \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_3 \end{bmatrix}$$

Equality constraints:

- Dynamics

$$[M(q) \quad -J^T(q) \quad -I] \begin{bmatrix} \ddot{q} \\ F \\ \tau \end{bmatrix} = -h(q, \dot{q})$$

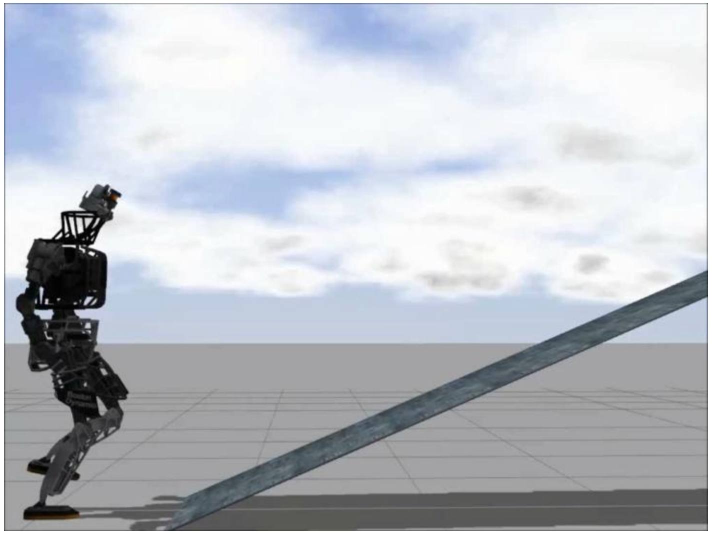
Inverse kinematics

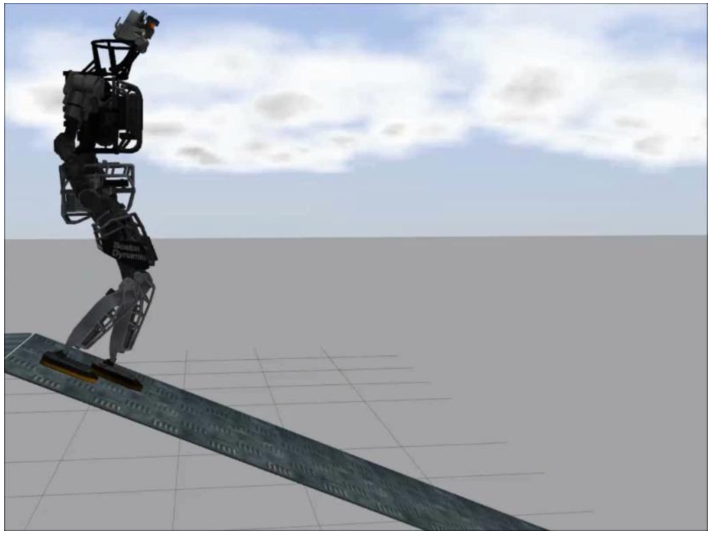
- Maintains its own state q_{ik}
- $\mathcal{X} = \dot{q}_{ik}$
- Integrate to get q_{ik}
- Uses desired motions from high level controller and internal states to compute target velocity with feedback
  $\dot{x}^* = K_{ik}(x_d - x) + \dot{x}_d$

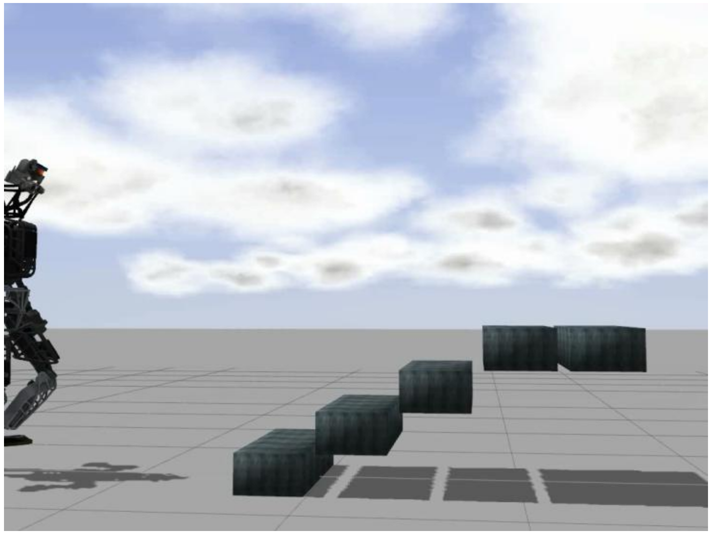
State estimation

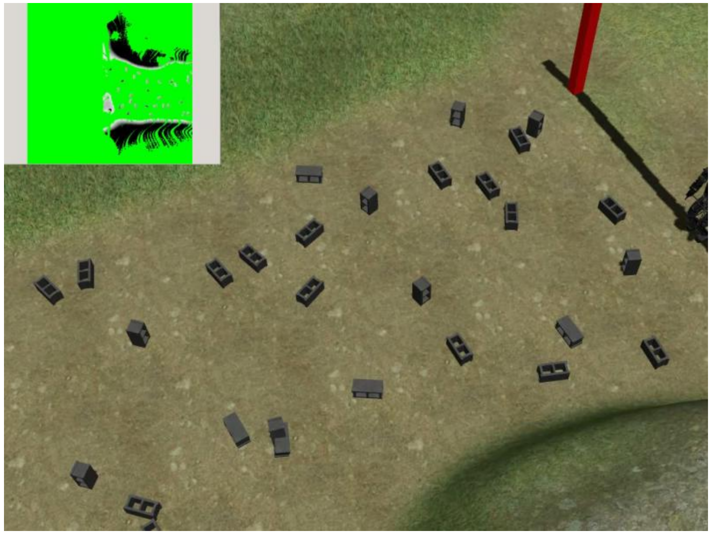
- Pelvis position and velocity are estimated with EKF.
  - Process model: IMU’s acceleration measurement
  - Observation model: FK results assuming known stationary contacts
- Pelvis orientation comes directly from IMU.
- Low pass filter joint velocities

Simulated walking









Optimizing CoM trajectory

- Trajectory optimization with a point mass model using Differential Dynamic Programming (DDP)
- Generalize to nonlinear models

$$X = (x, y, z, \dot{x}, \dot{y}, \dot{z})$$

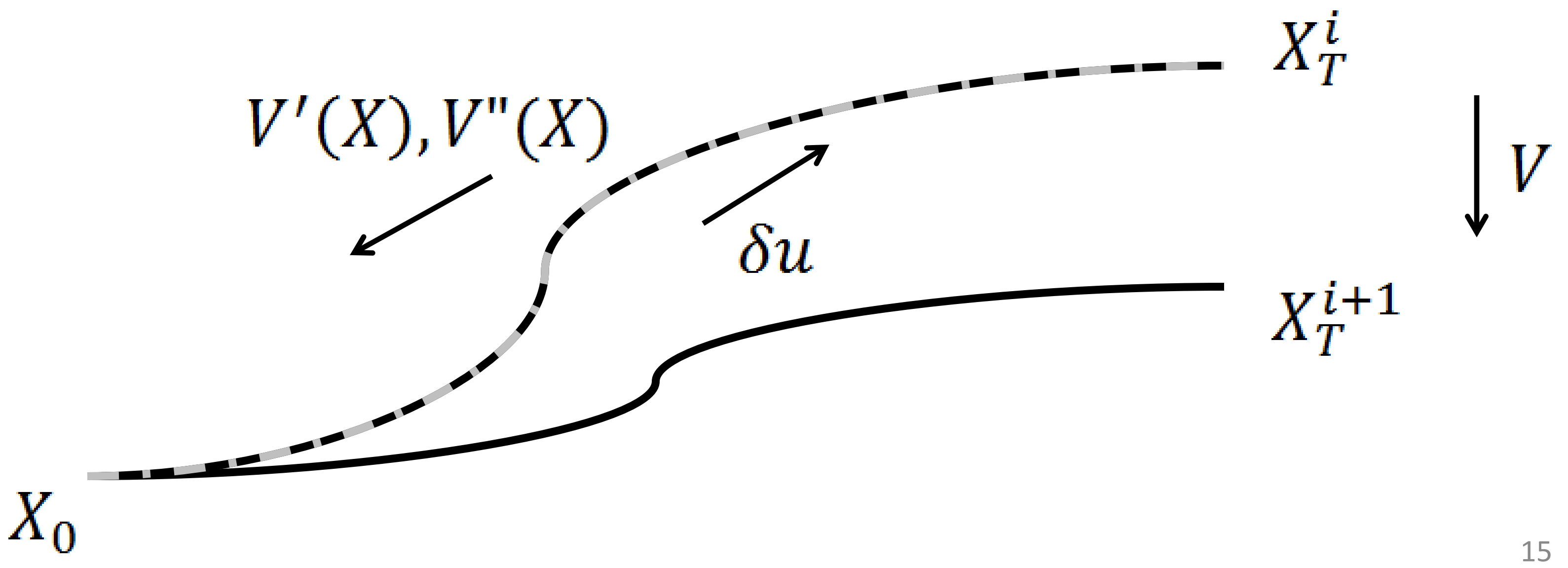
$$u = (p_x, p_y, F_z)$$

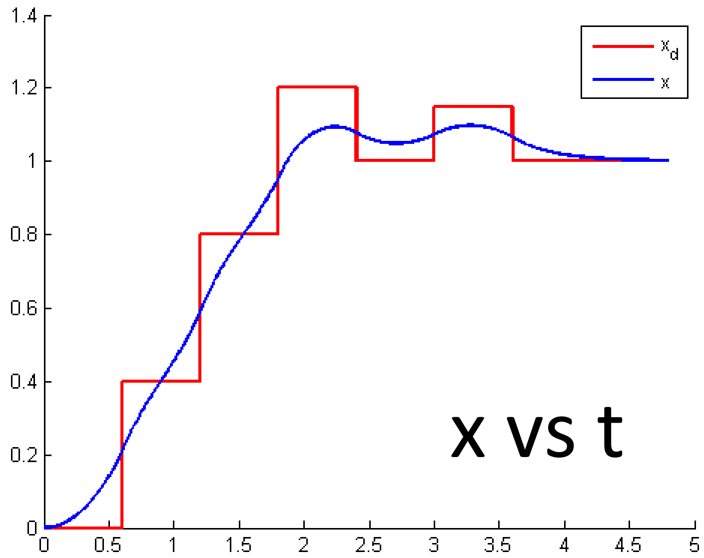
$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \dfrac{(x - p_x)F_z}{mz} \\ \dfrac{(y - p_y)F_z}{mz} \\ \dfrac{F_z}{m} - g \end{bmatrix}$$

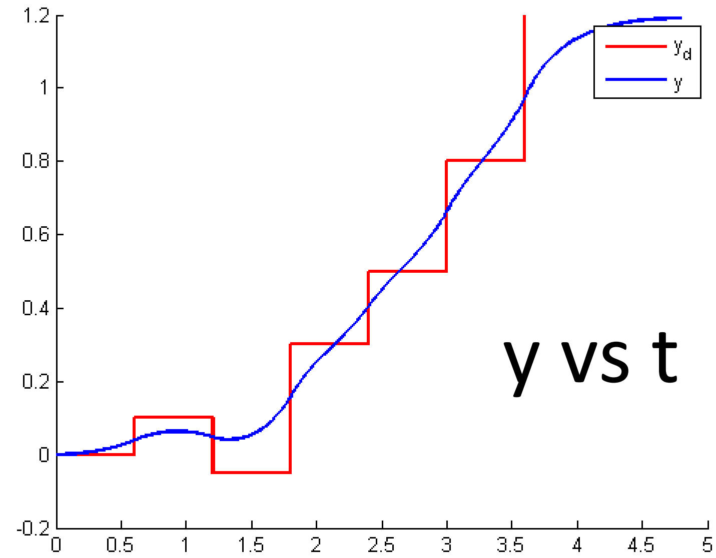
DDP

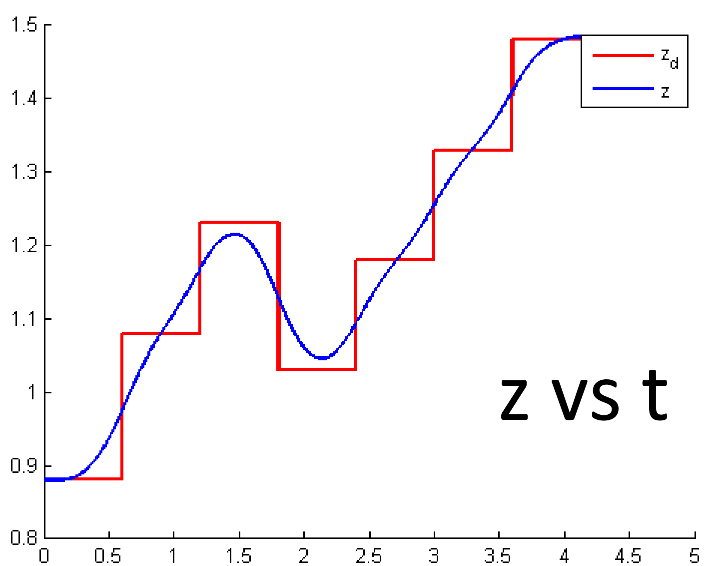
- One step cost function

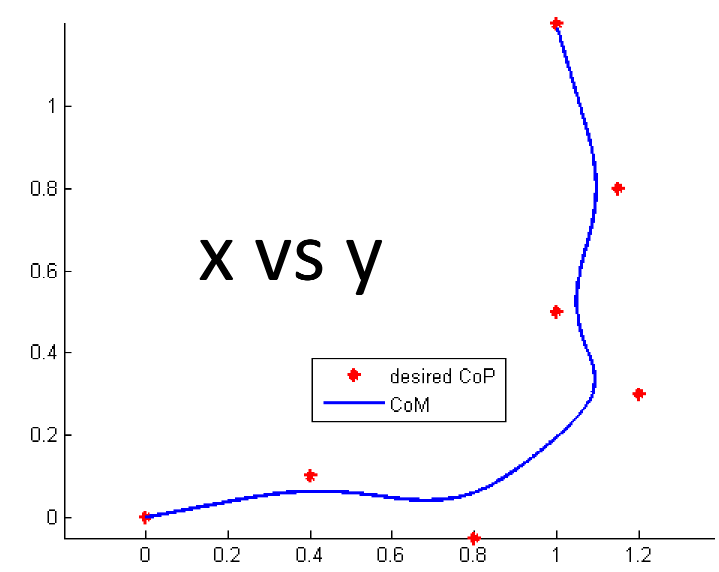
$$L(X,u) = 0.5(X - X^*)^T Q(X - X^*) \\ +0.5(u - u^*)^T R(u - u^*)$$




x vs t
x_d
x
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5


y vs t
y_d
y
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

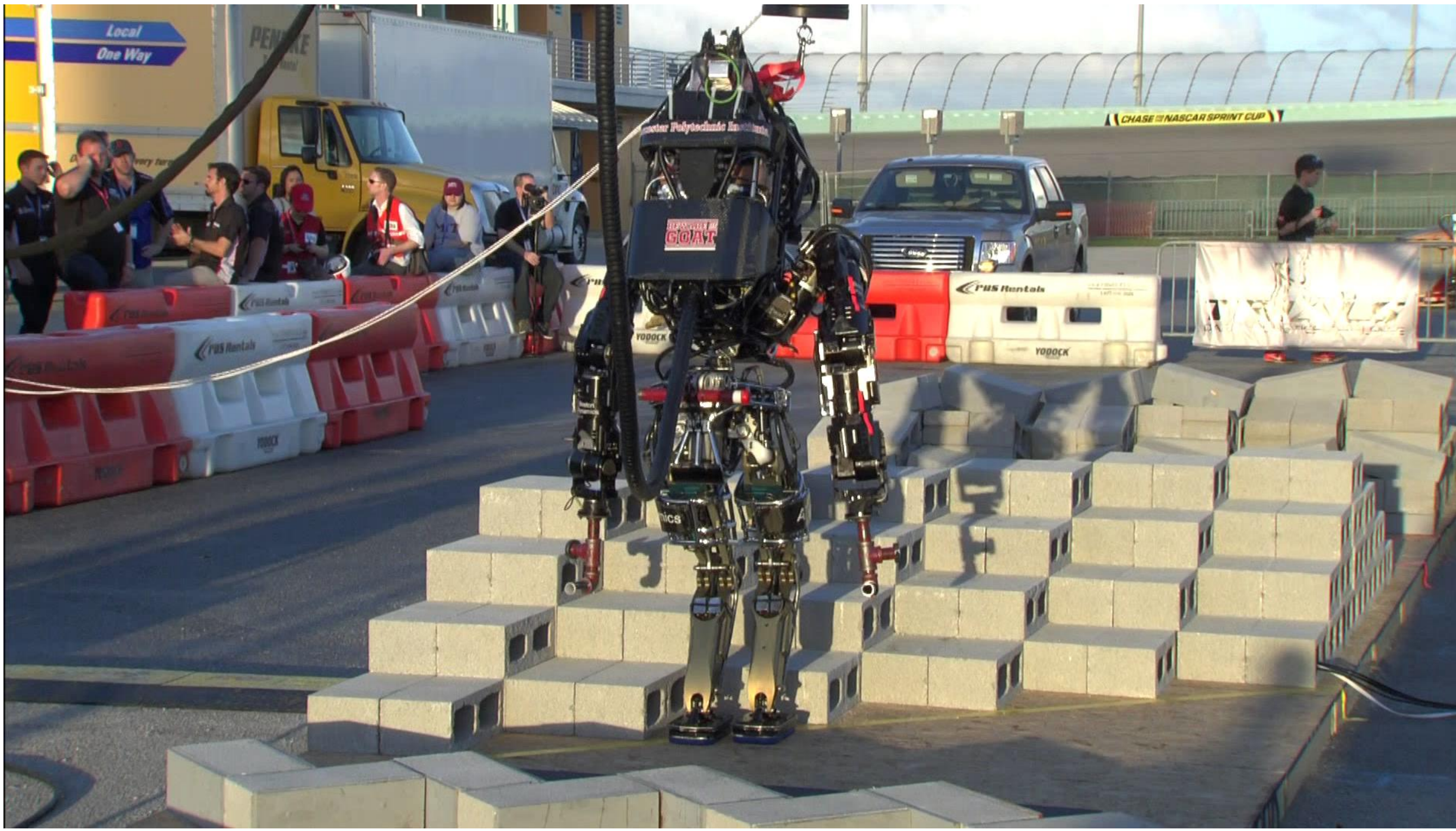

z vs t
z_d
z
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5


x vs y
desired CoP
CoM
1
0.8
0.6
0.4
0.2
0
0 0.2 0.4 0.6 0.8 1 1.2

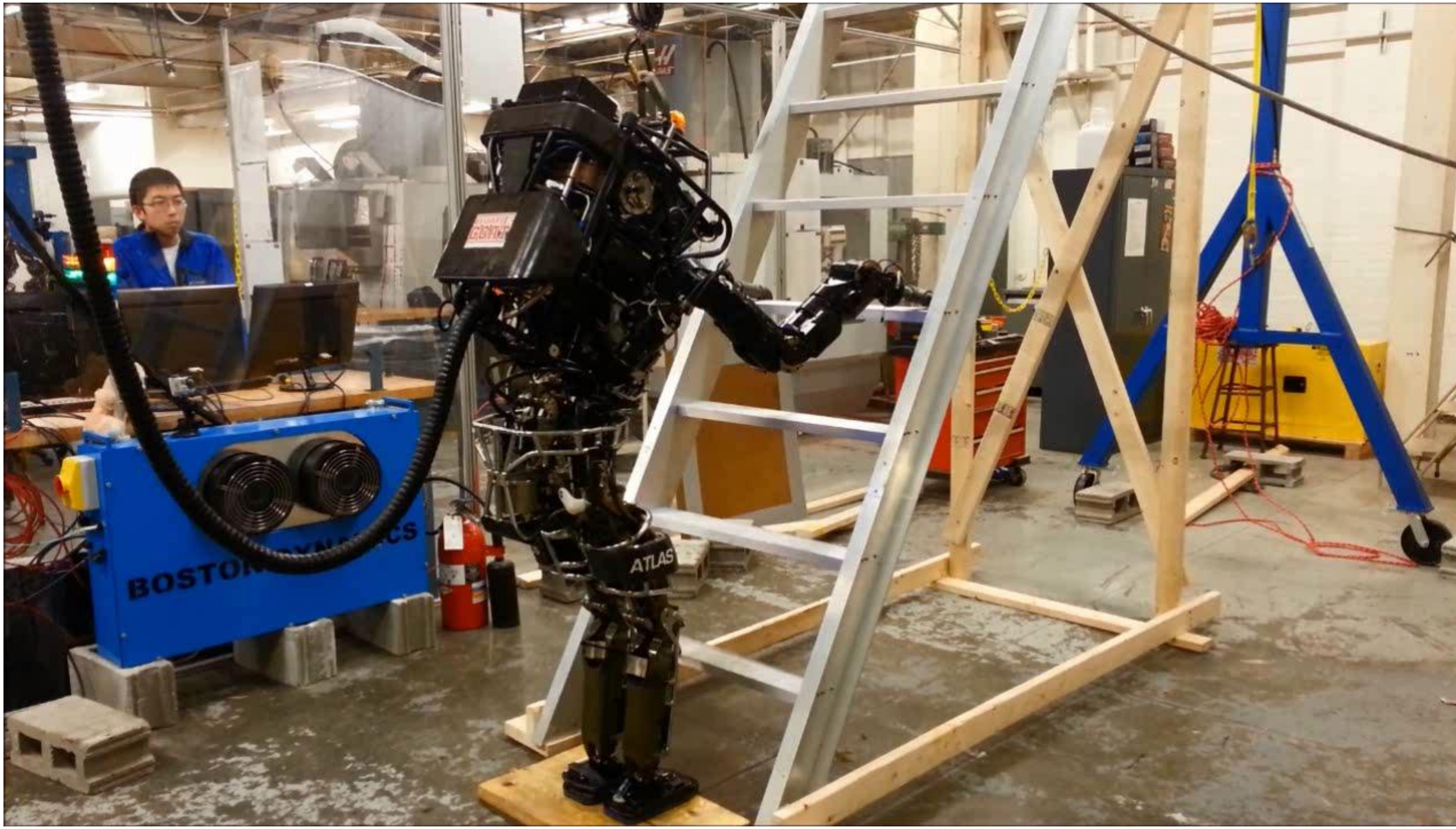
Implementing the DRC Trails tasks on Atlas

- Human in the loop “visual” servoing
  - Manual foot step / hand hold selection
  - Desired targets are specified incrementally.
- Compensate modeling errors with integrators
- Static walking:
  - Toe off is necessary for the terrain task.
  - Stance ankle is torque controlled.
- Ladder climbing:
  - Use hook hands
  - Scripted climbing sequence
  - Intentionally lean on the railings to stop yaw rotation
- Manipulation:
  - Motion scripts
  - End effector tracking

Atlas static walking

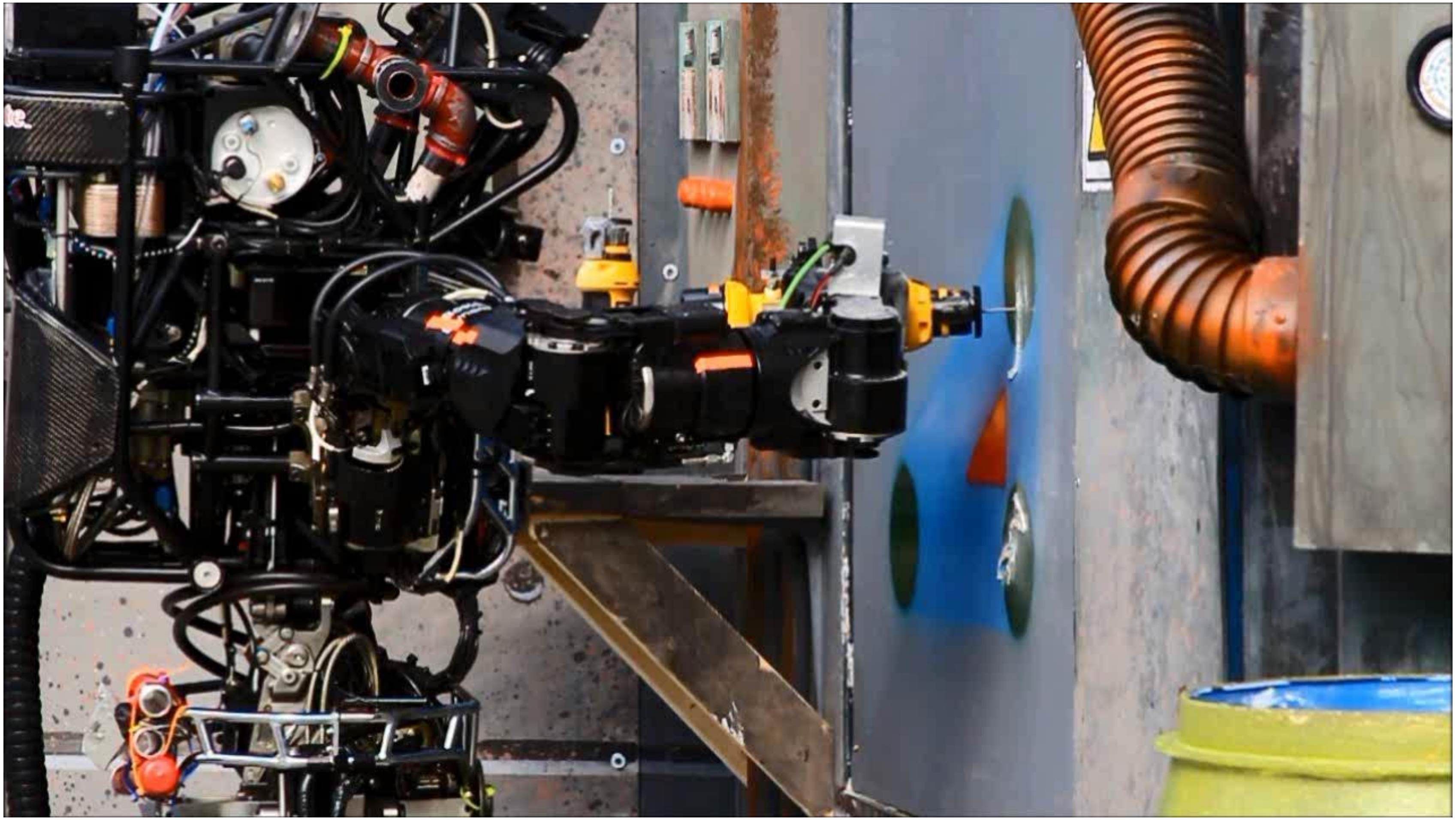


Atlas ladder climbing

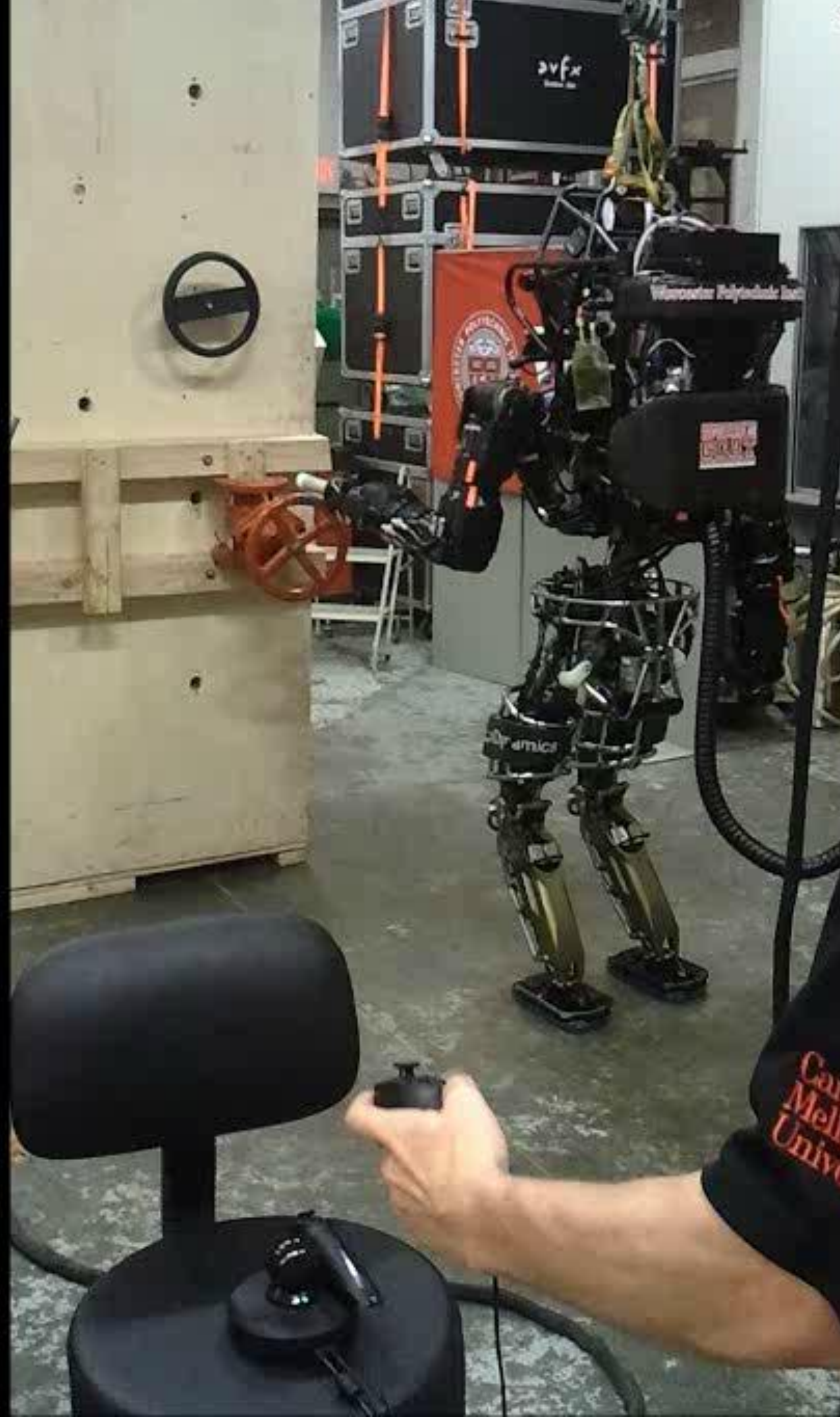


Eric Whitman

Atlas manipulation



Eric Whitman, Felipe Polido, Henrique Polidy



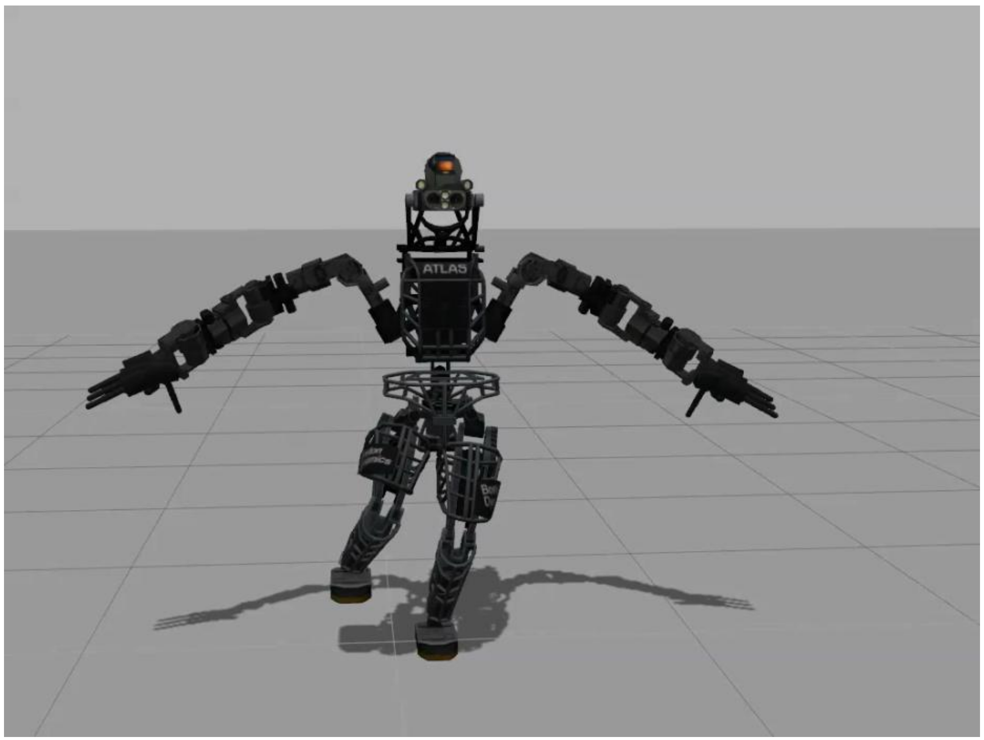
Remarks

- Controller
  - Divide and conquer
  - Go slow, use integrators
  - Inconsistency between IK and ID
- Atlas
  - Very repeatable
  - Arms have only 6 DoF, and are very weak.
  - Leg position and torque sensing are pre-transmission
    - Induce big FK errors.
    - Stiction degrades torque control.

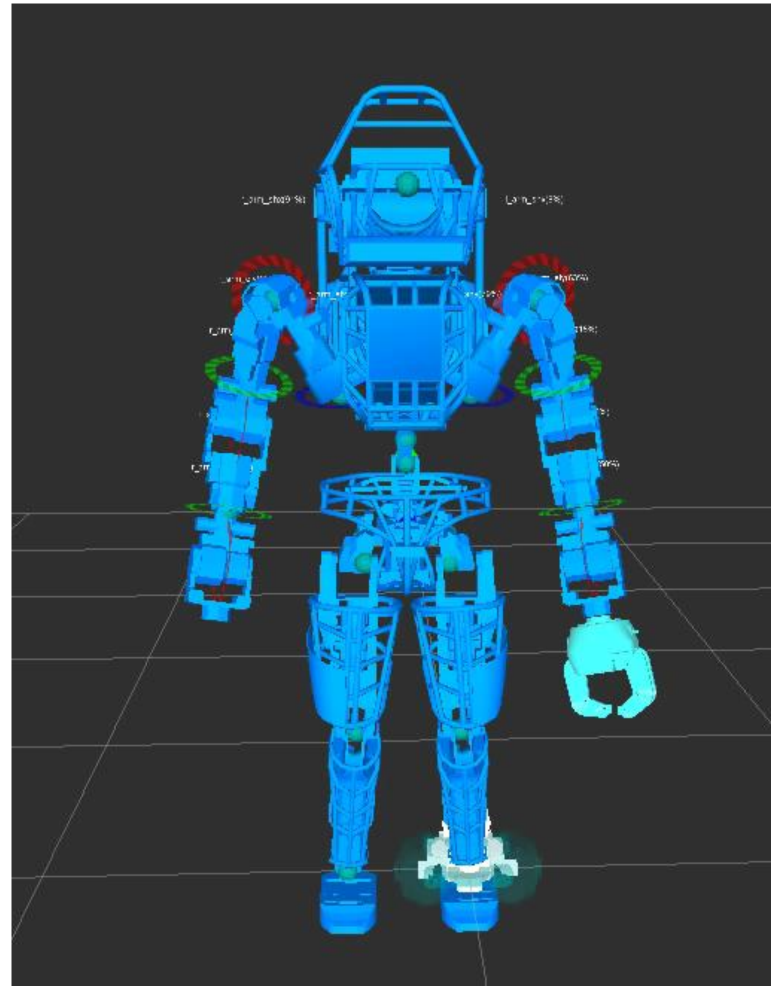
Work in progress

- Adding full body motion planning for manipulation
- Improve joint level servos
- Experimenting with integrating desired acceleration into desired velocity (replacing IK)
- Estimate modeling error online
- Adding angular momentum in the high level controller
- Introduce fixed delays in ID
- Self collision avoidance

Push = 20 Ns



Self collision avoidance



Felipe Polido

Questions?

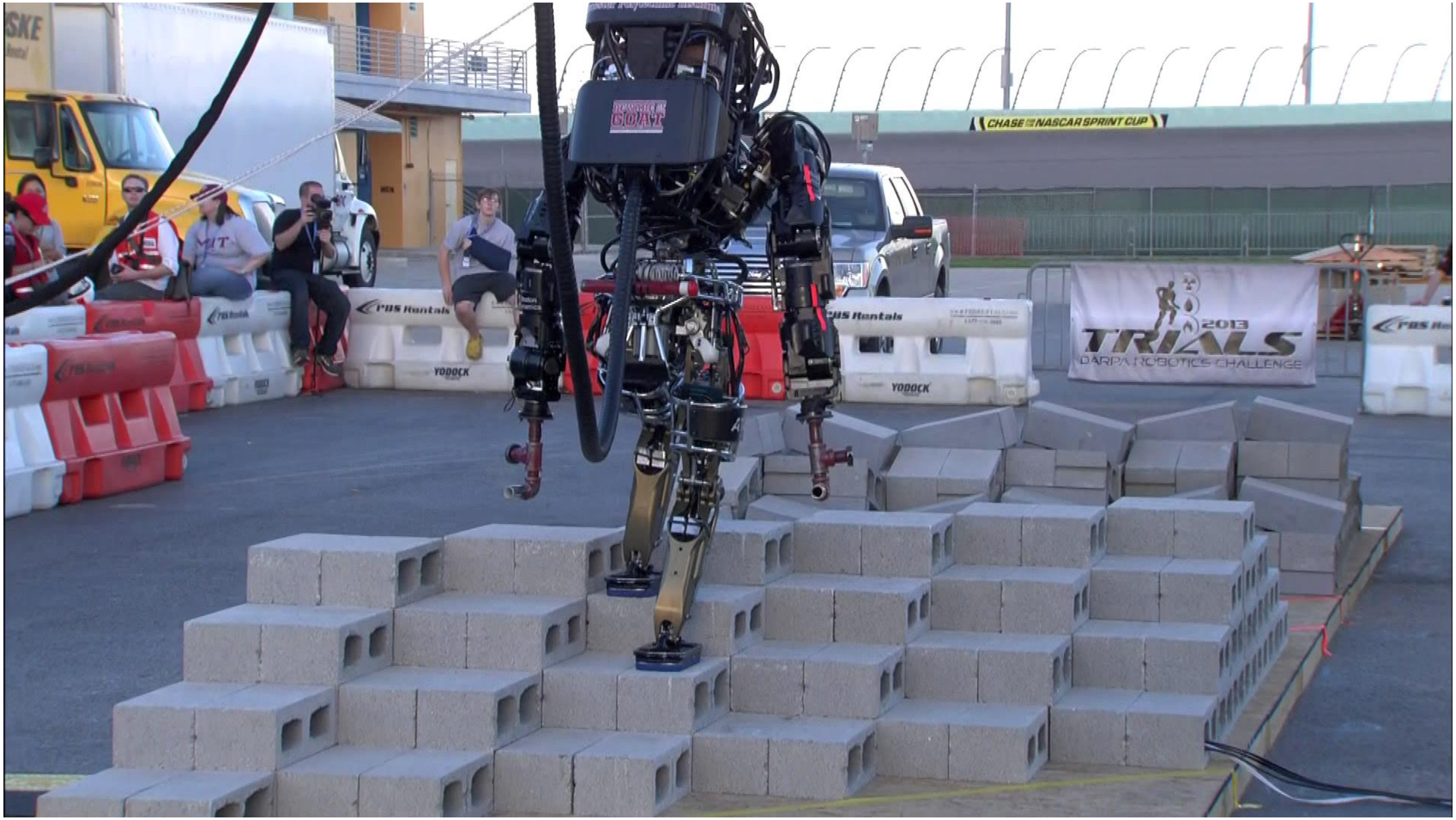
Dilemma: ID + IK

- Plan IK first, then do ID with IK's solution
  - Lose compliance at contacts
  - Vulnerable to unexpected perturbation
- Integrate ID's acceleration into velocity and position
  - Controller becomes unstable due to modeling error and delays.

Next step

- Build actuator model for better servo performance
- High level controller
  - Add angular momentum in the simple models
  - Re-optimize step timing and location
- Low level controller
  - Model modeling errors / better state estimation
  - Coordinate IK and ID better
  - Incorporate value functions in the QPs
  - Optimize valve command in ID

Atlas static walking



Atlas static walking