An Operational Method Toward Efficient Walk Control Policies for Humanoid Robots

Ludovic Hofer, Quentin Rouxel
ludovic.hofer@labri.fr, quentin.rouxel@labri.fr

Problem Definition

Humanoid robots playing soccer

Problem: Approach the ball and kick:

- Drive the walk to place the robot in kick position
- Align with goal posts orientation
- Do not touch the ball
- On holonomic robot (HA)
- On almost non holonomic robot (ANHA)

Action space (dimension = 3): Walk is controlled in acceleration.

Name	Units	min	max
Forward acc	$\frac{m}{\text{step}^2}$	-0.02	0.02
Lateral acc	$\frac{m}{\text{step}^2}$	-0.01	0.01
Angular acc	$\frac{\text{rad}}{\text{step}^2}$	-0.15	0.15

State space (dimension = 6):

Name	Units	min	max
Ball distance	m	0	1
Ball direction	rad	$-\pi$	π
Kick direction	rad	$-\pi$	π
Forward speed	$\frac{m}{\text{step}}$	-0.02	0.04
Lateral speed	$\frac{m}{\text{step}}$	-0.02	0.02
Angular speed	$\frac{\text{rad}}{\text{step}}$	-0.2	0.2

Problem features:

- Continuous action space
- Continuous state space
- Stochastic displacement model
- Discontinuous reward

Compared Policies

Winner2016: Expert policy used by Rhoban team to win RoboCup 2016. Manually tunned parameters.

CMA-ES: Same as Winner2016 with parameters tunned using black-box optimization CMA-ES in simulation.

RFPI: Policy represented as regression forest. Computed using MDP *Random Forest Policy Iteration* (RTFI) solver.

Proposed Method

1. Learn displacement model from data
2. Solve MDP problem in simulation
3. Apply on real robot

Displacement Model Learning

- Large discrepancies between walk orders and actual displacement
- Use data to learn corrective linear model
- No need for external hardware

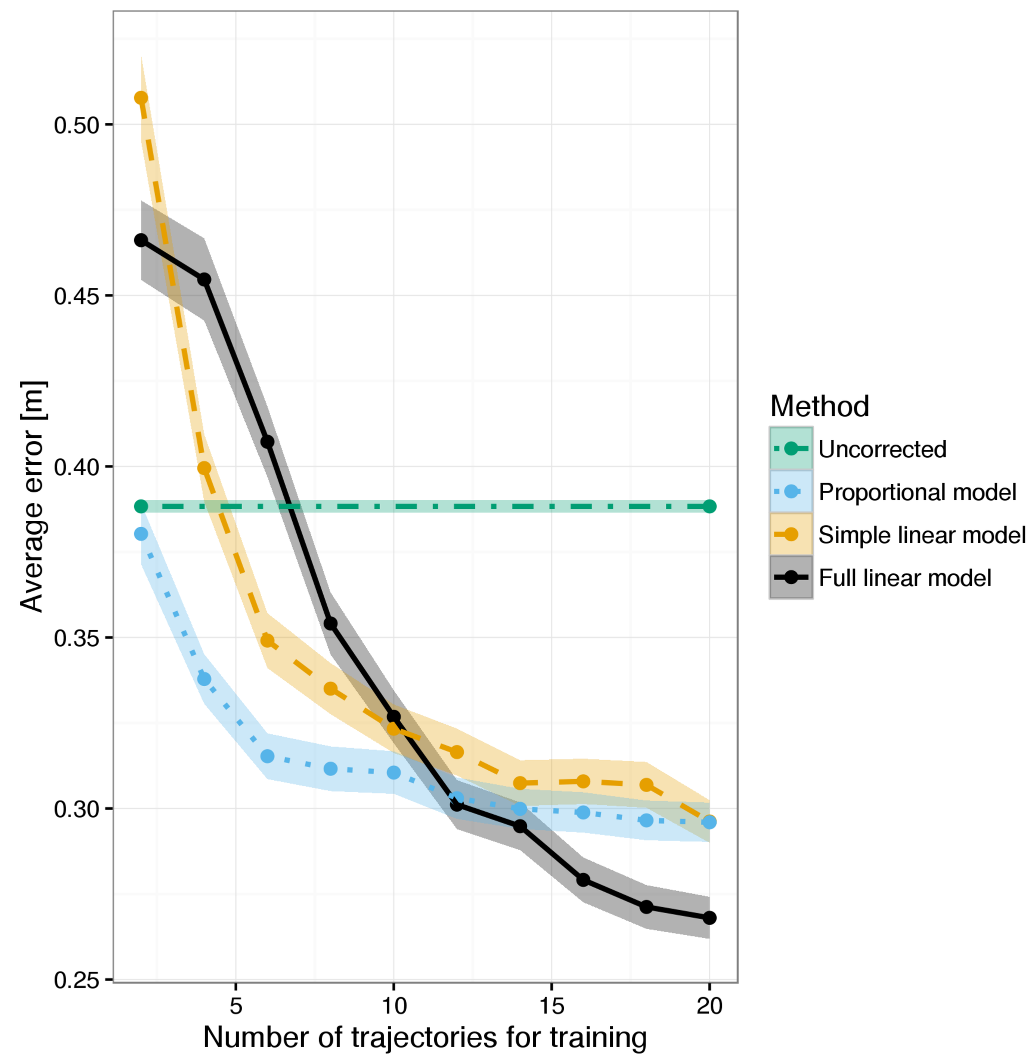
$$\Delta(x, y, \theta)_{\text{walk orders},k} \longmapsto \Delta(x, y, \theta)_{\text{corrected},k}$$

Compare different linear models:

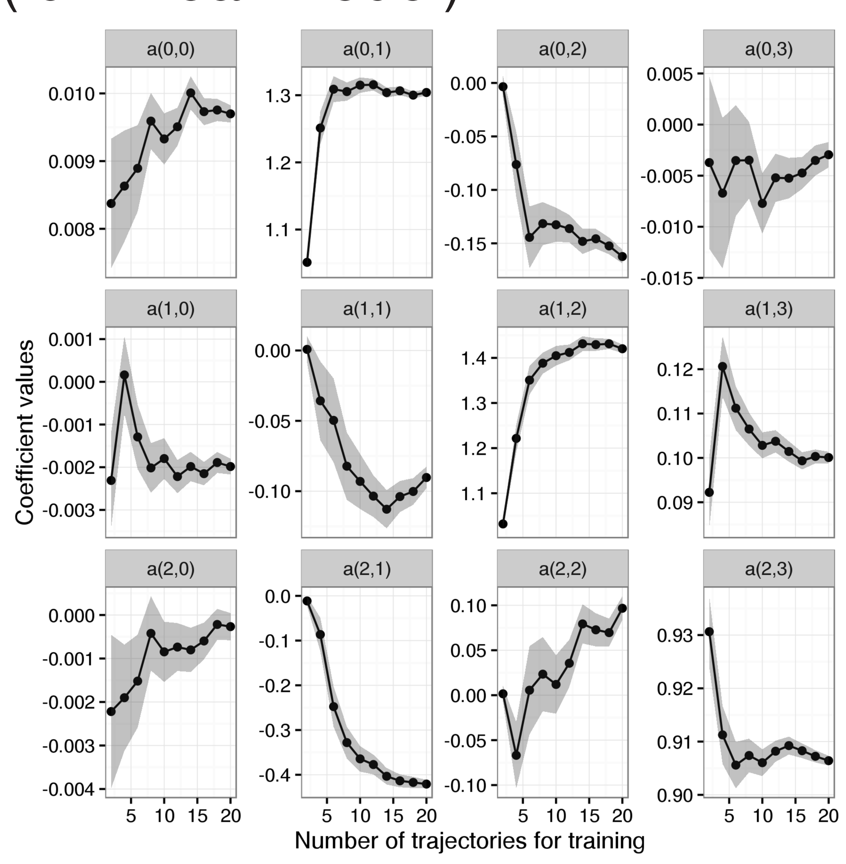
$$\begin{bmatrix} \Delta x_{corrected} \\ \Delta y_{corrected} \\ \Delta \theta_{corrected} \end{bmatrix} = M \begin{bmatrix} 1 \\ \Delta x_k \\ \Delta y_k \\ \Delta \theta_k \end{bmatrix}$$

- Proportional Model: 3 parameters.
- Simple Linear Model: 6 parameters.
- Full Linear Model: 12 parameters.

Parameter learning through black-box (CMA-ES) optimization with 20 learning data sequences:



Model parameters convergence:
(full linear model):



Contributions

- MDP continuous state and action solver (random forests)
- Displacement model learning procedure without external hardware
- Real robot applications
- Approach time improved on holonomous and non holonomous humanoid robot

MDP RFPI Solver

The CSA-MDP learner algorithm:

```
π = getRandomPolicy()
V = buildConstantApproximator(0)
visitedStates = seedStates
policyId = 1
runId = 0
while timeRemaining() do
    executeRun(π, visitedStates)
    runId++
    if runId == policyId then
        // Perform roll outs from visited states and
        // fit a regression forest with piecewise
        // constant model approximators.
        V = updateValue(π, V, visitedStates)
        // For every visited state: 1-step
        // optimization of action.
        // Then fit a regression forest with
        // piecewise linear model approximators.
        π = updatePolicy(V, visitedStates)
        runId = 0
        policyId++
    end if
end while
```

Results

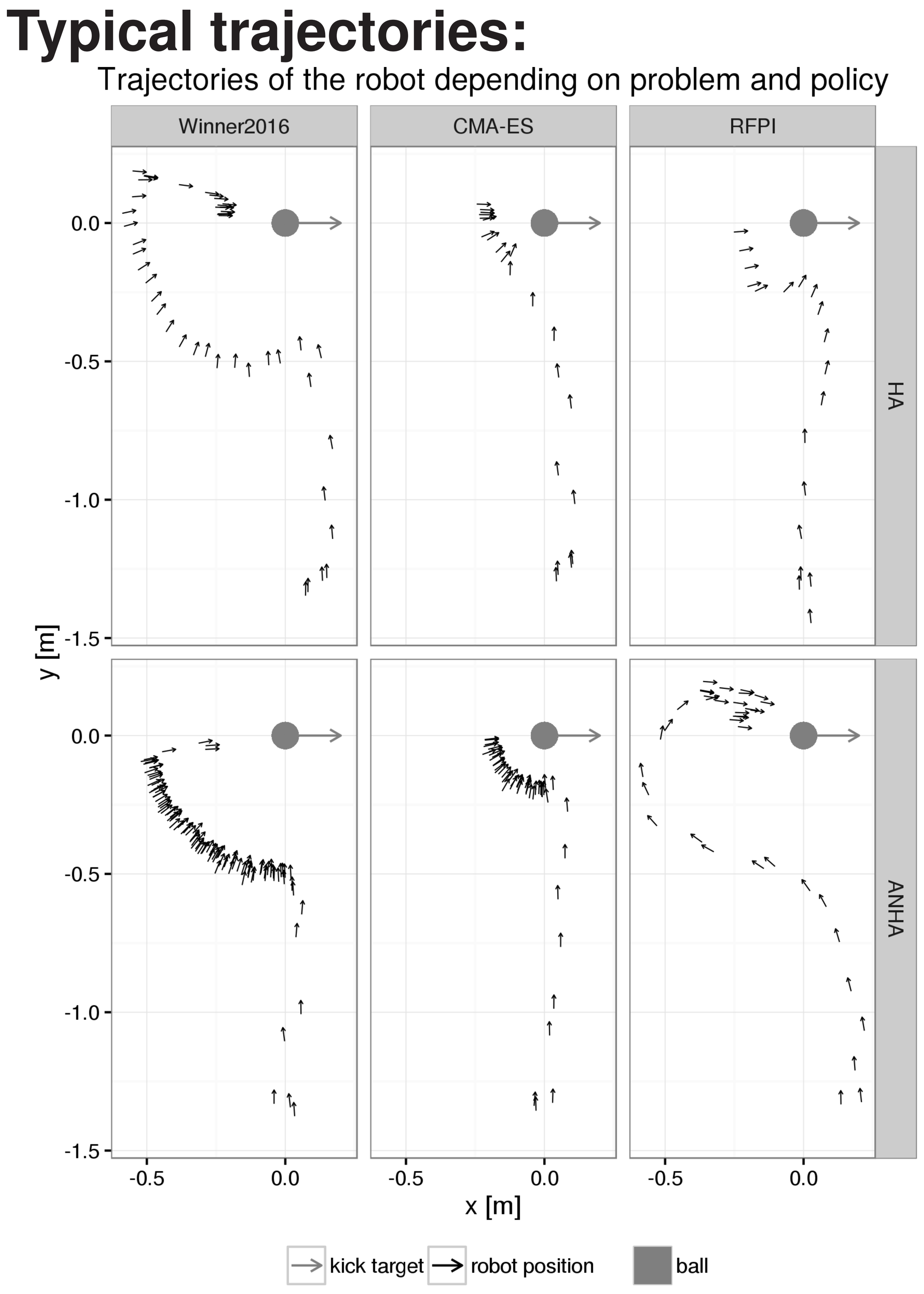
In simulation: average fitness costs:

	Winner2016	CMA-ES	RFPI
HA	31.84	14.90	11.88
ANHA	44.12	36.18	15.97

On physical robot: average time in seconds before kicking the ball:

	Winner2016	CMA-ES	RFPI
HA	19.98	13.72	11.45
ANHA	48.14	25.69	18.81

Typical trajectories:



References

- Hofer, L., and Gimbert, H. Online reinforcement learning for real-time exploration in continuous state and action markov decision processes. In PlanRob2016, ICAPS 2016.
- Rouxel, Q.; Passault, G.; Hofer, L.; N'Guyen, S.; and Ly, O. Learning the odometry on a small humanoid robot. In ICRA 2016.
- Ernst, D.; Geurts, P.; and Wehenkel, L. Tree-Based Batch Mode Reinforcement Learning. In JMLR 2005.