Using Dynamic Rewards to Learn a Fully Holonomic Bipedal Walk

Patrick MacAlpine and Peter Stone

Department of Computer Science, The University of Texas at Austin

June 4, 2012
Definitions

- **Bipedal locomotion**: Walking upright on two legs

- **Fully holonomic**: Able to move in all directions with equal velocity
RoboCup 3D Simulation Domain

- Teams of 9 vs 9 autonomous agents play soccer
- Realistic physics using Open Dynamics Engine (ODE)
- Agents modeled after Aldebaron Nao robot
- Agent receives noisy visual information about environment
- Agents can communicate with each other over limited bandwidth channel
## Competition Results

<table>
<thead>
<tr>
<th>RoboCup</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals For:</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Goals Against:</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Record (W-L-T):</td>
<td>4-5-1</td>
<td></td>
</tr>
<tr>
<td>Place:</td>
<td>Outside Top-8</td>
<td></td>
</tr>
</tbody>
</table>
## Competition Results

<table>
<thead>
<tr>
<th>RoboCup</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goals For:</strong></td>
<td>11</td>
<td>136</td>
</tr>
<tr>
<td><strong>Goals Against:</strong></td>
<td>17</td>
<td></td>
</tr>
<tr>
<td><strong>Record (W-L-T):</strong></td>
<td>4-5-1</td>
<td></td>
</tr>
<tr>
<td><strong>Place:</strong></td>
<td></td>
<td>Outside Top-8</td>
</tr>
</tbody>
</table>

*BIG IMPROVEMENT!* Optimized omnidirectional walk propelled team from 10th to 1st place.
## Competition Results

<table>
<thead>
<tr>
<th>RoboCup</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals For:</td>
<td>11</td>
<td>136</td>
</tr>
<tr>
<td>Goals Against:</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Record (W-L-T):</td>
<td>4-5-1</td>
<td></td>
</tr>
<tr>
<td>Place:</td>
<td>Outside Top-8</td>
<td></td>
</tr>
</tbody>
</table>
## Competition Results

<table>
<thead>
<tr>
<th>RoboCup</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals For:</td>
<td>11</td>
<td>136</td>
</tr>
<tr>
<td>Goals Against:</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Record (W-L-T):</td>
<td>4-5-1</td>
<td>24-0-0</td>
</tr>
<tr>
<td>Place:</td>
<td>Outside Top-8</td>
<td></td>
</tr>
</tbody>
</table>

*BIG IMPROVEMENT!* Optimized omnidirectional walk propelled team from 10th to 1st

Patrick MacAlpine (2012)
## Competition Results

<table>
<thead>
<tr>
<th>RoboCup</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals For:</td>
<td>11</td>
<td>136</td>
</tr>
<tr>
<td>Goals Against:</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Record (W-L-T):</td>
<td>4-5-1</td>
<td>24-0-0</td>
</tr>
<tr>
<td>Place:</td>
<td>Outside Top-8</td>
<td>1st</td>
</tr>
</tbody>
</table>

**BIG IMPROVEMENT!**

Optimized omnidirectional walk propelled team from 10th to 1st.
## Competition Results

<table>
<thead>
<tr>
<th>RoboCup</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals For:</td>
<td>11</td>
<td>136</td>
</tr>
<tr>
<td>Goals Against:</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Record (W-L-T):</td>
<td>4-5-1</td>
<td>24-0-0</td>
</tr>
<tr>
<td>Place:</td>
<td>Outside Top-8</td>
<td>1st</td>
</tr>
</tbody>
</table>

**BIG IMPROVEMENT!**
<table>
<thead>
<tr>
<th></th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals For:</td>
<td>11</td>
<td>136</td>
</tr>
<tr>
<td>Goals Against:</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Record (W-L-T):</td>
<td>4-5-1</td>
<td>24-0-0</td>
</tr>
<tr>
<td>Place:</td>
<td>Outside Top-8</td>
<td>1st</td>
</tr>
</tbody>
</table>

**BIG IMPROVEMENT!**

Optimized omnidirectional walk propelled team from 10th to 1st
Omnidirectional Walk Engine

- Double linear inverted pendulum model
- Based closely on that of walk engine by Graf et al
- Mostly open loop but not entirely
- Designed on actual Nao robot
## Walk Engine Parameters

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{maxStep}_i$</td>
<td>Maximum step sizes allowed for $x$, $y$, and $\theta$</td>
</tr>
<tr>
<td>$y_{\text{shift}}$</td>
<td>Side to side shift amount with no side velocity</td>
</tr>
<tr>
<td>$z_{\text{torso}}$</td>
<td>Height of the torso from the ground</td>
</tr>
<tr>
<td>$z_{\text{step}}$</td>
<td>Maximum height of the foot from the ground</td>
</tr>
<tr>
<td>$f_g$</td>
<td>Fraction of a phase that the swing foot spends on the ground before lifting</td>
</tr>
<tr>
<td>$f_a$</td>
<td>Fraction that the swing foot spends in the air</td>
</tr>
<tr>
<td>$f_s$</td>
<td>Fraction before the swing foot starts moving</td>
</tr>
<tr>
<td>$f_m$</td>
<td>Fraction that the swing foot spends moving</td>
</tr>
<tr>
<td>$\phi_{\text{length}}$</td>
<td>Duration of a single step</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Factors of how fast the step sizes change</td>
</tr>
<tr>
<td>$y_{\text{sep}}$</td>
<td>Separation between the feet</td>
</tr>
<tr>
<td>$x_{\text{offset}}$</td>
<td>Constant offset between the torso and feet</td>
</tr>
<tr>
<td>$x_{\text{factor}}$</td>
<td>Factor of the step size applied to the forwards position of the torso</td>
</tr>
<tr>
<td>$\text{err}_{\text{norm}}$</td>
<td>Maximum COM error before the steps are slowed</td>
</tr>
<tr>
<td>$\text{err}_{\text{max}}$</td>
<td>Maximum COM error before all velocity reach 0</td>
</tr>
</tbody>
</table>

Parameters of the walk engine with the optimized parameters shown in bold
Initial Walk Parameters

- Designed and hand-tuned to work on the actual Nao robot
- Provides a slow and stable walk
CMA-ES (Covariance Matrix Adaptation Evolutionary Strategy)

- **Evolutionary** numerical optimization method
- Candidates sampled from multidimensional Gaussian and evaluated for their fitness
- Weighted average of members with highest fitness used to update mean of distribution
- Covariance update using *evolution paths* controls search step sizes

Patrick MacAlpine (2012)
CMA-ES (Covariance Matrix Adaptation Evolutionary Strategy)

- **Evolutionary** numerical optimization method
- Candidates sampled from multidimensional Gaussian and evaluated for their fitness
- Weighted average of members with highest fitness used to update mean of distribution
- Covariance update using *evolution paths* controls search step sizes
Learning Algorithms Evaluation

- **CEM**: Cross Entropy Method
- **CMA-ES**: Covariance Matrix Strategy Evolutionary Strategy
- **GA**: Genetic Algorithm
- **HC**: Hill Climbing
- **RWG**: Random Weight Guessing

Patrick MacAlpine (2012)
Agent moves and turns in direction of target at the same time
When dribbling agent circles while always facing ball
Learn three different parameter sets for three different tasks
  - Going to a target
  - Sprinting forward
  - Positioning around the ball when dribbling

Parameters learned through a layered learning approach

I = initial, T = goToTarget, S = sprint, P = positioning
Agent navigates to a series of target positions on the field
Also have stop targets where agent is told to stop
Reward: + for distance traveled toward target,
- for movement when told to stop

\[
\text{Fall} = 5 \text{ if robot fell, 0 otherwise}
\]
\[
d_{\text{target}} = \text{distance traveled towards target}
\]
\[
d_{\text{moved}} = \text{total distance moved}
\]
\[
t_{\text{total}} = \text{duration a target is active}
\]
\[
t_{\text{taken}} = \text{time taken to reach target, or } t_{\text{total}} \text{ if target not reached}
\]

\[
\text{reward}_{\text{target}} = d_{\text{target}} \frac{t_{\text{total}}}{t_{\text{taken}}} - \text{Fall}
\]
\[
\text{reward}_{\text{stop}} = -d_{\text{moved}} - \text{Fall}
\]
Red ‘T’ = gotoTarget parameters, yellow ‘S’ = sprint parameters
Red ‘T’ = gotoTarget parameters, yellow ‘S’ = sprint parameters, cyan ‘P’ = positioning parameters
2011 Walk Weaknesses

- Still not all that fast moving around the ball
- Turning takes time and causes a delay
Fully Holonomic Walk

- Want to be able to walk in all directions with equal velocity
- No delays for needing to turn
Problems in Learning a Fully Holonomic Walk

- Kinematics of robot allow for faster walking forward speed
- Speed in one direction dominates speed in other directions
- Agent optimized without turning to target lost on average by .7 goals to agent that does turn
Fully Holonomic Walk Optimization

- Use GoToTarget optimization but agent does not turn toward target
- Only give positive rewards during longs walks in cardinal forward, backward, and sideways directions
- Still penalize for falls in all parts of the optimization
- Dynamically reweight directional rewards to encourage equal velocities in each direction

\[ reward = reward_{fw} \times weight_{fw} + reward_{bw} \times weight_{bw} + reward_{sw} \times weight_{sw} \]
Reweighting Rewards

Separate directional rewards from overall reward (from top fitness member or weighted average of top half of population)

\[ \text{reward}_i \Rightarrow \text{reward}_i \{ \text{fw}, \text{bw}, \text{sw} \} \]

Get maximum reward for any of the directions

\[ \text{reward}_i \{ \text{max} \} = \max (\text{reward}_i \{ \text{fw}, \text{bw}, \text{sw} \}) \]

Compute weights (factors) to multiply each directional reward by to equal maximum reward

\[ \text{weight}_i + 1 \{ \text{fw} / \text{bw} / \text{sw} \} = \text{reward}_i \{ \text{max} \} / \text{reward}_i \{ \text{fw} / \text{bw} / \text{sw} \} \]

Normalize all weights to sum to 1

\[ \text{weight}_i + 1 \{ \text{fw} / \text{bw} / \text{sw} \} = \text{weight}_i + 1 \{ \text{fw} / \text{bw} / \text{sw} \} / \sum (\text{weight}_i + 1 \{ \text{fw}, \text{bw}, \text{sw} \}) \]
Reweighting Rewards

Separate directional rewards from overall reward
(from top fitness member or weighted average of top half of population)

\[ reward_i \Rightarrow reward_i\{fw,bw,sw\} \]
Reweighting Rewards

Separate directional rewards from overall reward
(from top fitness member or weighted average of top half of population)

\[ \text{reward}_i \Rightarrow \text{reward}_{i\{fw,bw,sw\}} \]

Get maximum reward for any of the directions

\[ \text{reward}_{i\{max\}} = \max(\text{reward}_{i\{fw,bw,sw\}}) \]
Reweighting Rewards

Separate directional rewards from overall reward
(from top fitness member or weighted average of top half of population)

\[ reward_i \Rightarrow reward_i\{fw,bw,sw\} \]

Get maximum reward for any of the directions

\[ reward_i\{max\} = \max(reward_i\{fw,bw,sw\}) \]

Compute weights (factors) to multiply each directional reward by to equal maximum reward

\[ weight_{i+1}\{fw/bw/sw\} = reward_i\{max\}/reward_i\{fw/bw/sw\} \]
Reweighting Rewards

Separate directional rewards from overall reward (from top fitness member or weighted average of top half of population)

\[ \text{reward}_i \Rightarrow \text{reward}_i^{\{fw,bw,sw\}} \]

Get maximum reward for any of the directions

\[ \text{reward}_i^{\{\text{max}\}} = \max(\text{reward}_i^{\{fw,bw,sw\}}) \]

Compute weights (factors) to multiply each directional reward by to equal maximum reward

\[ \text{weight}_{i+1}^{\{fw/bw/sw\}} = \text{reward}_i^{\{\text{max}\}} / \text{reward}_i^{\{fw/bw/sw\}} \]

Normalize all weights to sum to 1

\[ \text{weight}_{i+1}^{\{fw/bw/sw\}} = \text{weight}_{i+1}^{\{fw/bw/sw\}} / \text{sum(\text{weight}_{i+1}^{\{fw,bw,sw\}})} \]
Weights Over Iterations of CMA-ES

- Both dynamic reward agent’s weights converge to almost the same value

- Static reward agent’s weights (not applied to reward) diverge as forward speed dominates
Directional Speeds

<table>
<thead>
<tr>
<th>Agent</th>
<th>Forward</th>
<th>Backward</th>
<th>Sideways</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynamicRewards</td>
<td>.42</td>
<td>.53</td>
<td>.48</td>
</tr>
<tr>
<td>DynamicAvgRewards</td>
<td>.45</td>
<td>.53</td>
<td>.51</td>
</tr>
<tr>
<td>StaticRewards</td>
<td>.58</td>
<td>.52</td>
<td>.37</td>
</tr>
<tr>
<td>FaceForward</td>
<td>.74</td>
<td>.35</td>
<td>.03</td>
</tr>
<tr>
<td>2011 Walk</td>
<td>.71</td>
<td>.40</td>
<td>.21</td>
</tr>
</tbody>
</table>

- Both dynamic reward agents have close to equal speeds in all directions.
- Static reward agent has slower side walking speed.
- Face forward agent very biased toward forward walking speed with almost 0 speed for sideways direction.
Fully Holonomic Walk

- Can walk in all directions with nearly equal velocity
Circling the Ball

Patrick MacAlpine (2012)
Reaction Time

Patrick MacAlpine (2012)
## Game Performance

<table>
<thead>
<tr>
<th></th>
<th>2011 Walk</th>
<th>FaceForward</th>
<th>StaticRewards</th>
<th>DynAvgRewards</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynRewards</td>
<td>0.20 (.08)</td>
<td>3.27 (.09)</td>
<td>3.18 (.11)</td>
<td>-0.06 (.07)</td>
</tr>
<tr>
<td>DynAvgRewards</td>
<td>0.10 (.07)</td>
<td>3.49 (.11)</td>
<td>2.88 (.11)</td>
<td></td>
</tr>
<tr>
<td>StaticRewards</td>
<td>-2.77 (.13)</td>
<td>0.22 (.06)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FaceForward</td>
<td>-2.99 (.12)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DynRewards vs 2011 Walk Record: 23-7-70 (29 goals for, 9 against)

Patrick MacAlpine (2012)
## Game Performance

<table>
<thead>
<tr>
<th></th>
<th>2011 Walk</th>
<th>FaceForward</th>
<th>StaticRewards</th>
<th>DynAvgRewards</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynRewards</td>
<td>0.20(.08)</td>
<td>3.27(.09)</td>
<td>3.18(.11)</td>
<td>-0.06(.07)</td>
</tr>
<tr>
<td>DynAvgRewards</td>
<td>0.10(.07)</td>
<td>3.49(.11)</td>
<td>2.88(.11)</td>
<td></td>
</tr>
<tr>
<td>StaticRewards</td>
<td>-2.77(.13)</td>
<td>0.22(.06)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FaceForward</td>
<td>-2.99(.12)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DynRewards vs 2011 Walk Record: 23-7-70 (29 goals for, 9 against)
Dynamically updating reward weights is an effective means for learning a fully holonomic walk. Rebalancing reward weights helps to prevent domination of one component of a reward signal over other components. In the 3D simulation league quickness is more important than speed.
Summary

- **Dynamically updating reward weights** is an effective means for learning a fully holonomic walk.
Summary

- Dynamically updating reward weights is an effective means for learning a fully holonomic walk.

- Rebalancing reward weights helps to prevent domination of one component of a reward signal over other components.
Dynamically updating reward weights is an effective means for learning a fully holonomic walk.

Rebalancing reward weights helps to prevent domination of one component of a reward signal over other components.

In the 3D simulation league quickness is more important than speed.
Related Work

Related Work

Future Work

- Attempt to apply learned walks in simulation to actual Nao robots
- Extend holonomic walk to use multiple parameter sets (one for each of the cardinal directions)
- Model walk trajectories after those taken by human infants learning to walk
More Information

UT Austin Villa 3D Simulation Team homepage:
www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/

Email: patmac@cs.utexas.edu

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030).
• All non-turning holonomic agents have similar fitness

• Face forward turning agent (similar to 2011 walk agent) has highest fitness
Average Weighted Rewards Calculation

\[
weight_i = \log \left( \frac{\text{popsizesize}}{2} + \frac{1}{2} \right) - \log(i)
\]

\[
weights_{\text{sum}} = \sum_{i=1}^{\text{popsizesize}/2} weight_i
\]

\[
weight_i = \frac{weight_i}{weights_{\text{sum}}}
\]

\[
rew_{\text{avg}}\{\text{fw/bw/sw}\} = \sum_{i=1}^{\text{popsizesize}/2} rew_i\{\text{fw/bw/sw}\} \times weight_i
\]