Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Learning Complex Stand-Up Motion for Humanoid Robots

Heejin Jeong and **Daniel D. Lee**
Department of Electrical and System Engineering
University of Pennsylvania
Philadelphia, PA 19104
{heejinj, ddlee}@seas.upenn.edu

Abstract

In order for humanoid robots to complete various assigned tasks without any human assistance, they must have the ability to stand up on their own. In this abstract, we introduce complex stand-up motion of humanoid robots learned by using Reinforcement Learning.

Introduction

In the current robotics field, there are a number of small-size humanoid robots with the ability to stand up after they fall down (Jörg Stuückler 2003). However, most of their stand-up motions consist of several motion sequences in a fixed order with specific sets of joint angles. In these cases, since a robot always has to move to the desired pose of the first sequence regardless of its initial pose after falling, such stand-up motions can lead to time delays in standing or cause damage to the robots from hitting the ground or receiving high torque forces. Adult-size humanoid robots can become even more seriously damaged by these motions since they are much heavier and have more servo motors than small-size humanoid robots in general. During the Defense Advanced Research Project Agency (DARPA) Robotics Challenge Final in 2015, many robots fell down multiple times, but only one robot managed to stand up. This shows that the ability to stand up from various poses in different environments becomes increasingly important to robots as they start to work in human environments or alongside humans.

In this research, we studied stand-up motions of a humanoid robot using reinforcement learning (RL). As an initial approach, we considered an obstacle-free environment with flat and even ground, and we discussed its scalability to complicated environment in the last section. We used a DARWIN-OP robot, which already has a hand-designed stand-up motion (*HS*), for the application platform. An optimal policy was learned in Webot simulation program. The policy has been applied to the real robot and we showed that the motion generated by the policy has a better result than previously hand designed motions.

Elements of RL for Stand-up Motions

Similar to humans, we considered stand-up situations of humanoid robots into two cases - standing up from body positions with chest down, right or left lateral recumbent (*case 1*), and from body positions with back down(*case 2*).

Constructing A Discrete State Space using EM

One of the challenges faced in applying RL to robotics is high-dimensional and continuous state space. Several methods have been used to define a state in a continuous domain such as the Fourier basis method (George Konidaris 2011). However, humanoid robots have a large number of joints, and thus a state space defined by such methods still has high dimensionality. In this research, we defined a continuous state space with a continuous state vector, $\mathbf{u}$, consisting of its joint positions ($q_i \in [-\pi, \pi]$, $i = 1, \cdots, 18$) and its body roll and pitch ($\varphi, \theta \in [-\pi, \pi]$). Then we discretized the space using one of the clustring methods, *Expectation-Maximization* (EM) algorithm for Gaussian Mixtures.

Using the EM method with initial spherical covariance and priors in uniform frequency, we constructed 30 clusters from continuous state data obtained by generating random motions from different falling down poses of the robot. Including each of motion sequences in *HS* as a cluster, we defined a discrete state space, S, of the model with two subspaces, S_1 and S_2, corresponding to *case1* and *case2*, respectively. Two finalization states are in $S_1 \cap S_2$ so that they can be reached in the both cases. At each step, a robot perceives its state $s^{(i)} \in S$ from a continuous state vector, $\mathbf{u}_t$ as follow:

$$s_t = s^{(i)} \quad \text{where } i = \text{argmax}_{1 \le k \le N_s} \|\mathbf{z}_k - \mathbf{u}_t\|_2$$

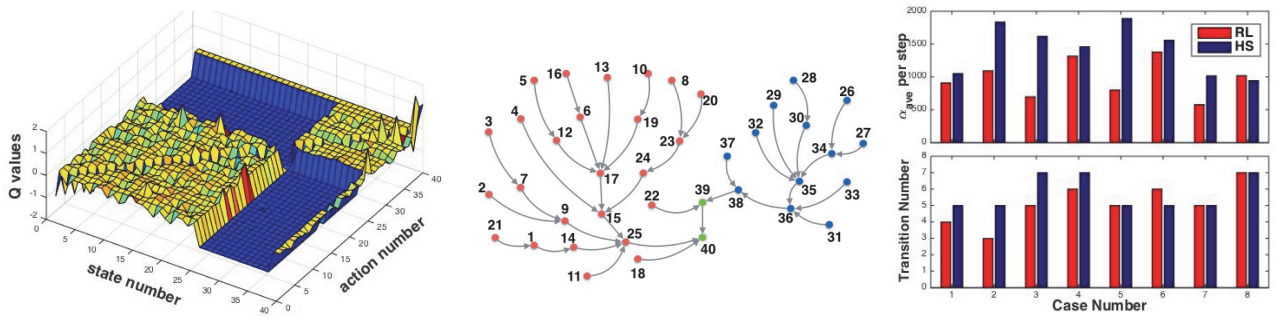
where $\mathbf{z_k}$ is the centroid vector of kth cluster and $N_s = 40$ is the number of discrete states.

Action

We define a discrete action space, A, with $a_k \in A$ ($k = 1, \cdots, N_s$) corresponding to the clusters. It also has two subspaces, A_1 and A_2, the s ame as the state space. Joint position values of each centroid vector of the clusters defines desired joint positions in each action.

Reward

In order to design an appropriate reward function, we considered the following five factors as the variables of the func-



(a) (b) (c)

Figure 1: (a) Final Q values; (b) State-Action Planning Map (red: *case1*, blue: *case2*, green: finalization); (c) Comparison between the stand-up motion learned by RL and the previously hand-designed motions(*HS*)

tion (at time step, t): Cumulated difference of body angular acceleration, Difference of current body attitude from desired attitude, Body height change ($H_t - H_{t-1}$), Body pitch change ($p_t - p_{t-1}$), and Body roll change ($r_t - r_{t-1}$). Then the hyperbolic tangent function is used for each variable, x_i, with different constant parameters (w_i, c_i, ν_i and b_i) to saturate at larges values and to exponentially increase or decrease at other values (Line 19 in table 1). The parameters were determined by considering trade-off among the effects of the variables.

Learning Algorithm

Q-learning is used ($\gamma = 0.9$) for the learning method. In RL, especially in a robotics system with relatively high dimensional state and action space, an exploration strategy plays a key role in reducing unnecessary exploring time and preventing a robot from being damaged. Therefore, we restricted the exploration area of a state $s \in S_l$ as A_l (table 1) and assigned different initial $Q(s \in S_i, a \in A_j)$ to the different cases: 0.01 for $i = j$, -0.1 for $i \neq j$, and 0.1 for $i = j = 1 \cap 2$. The numerical values were determined according to the range of reward values and the learning rate:

$$\alpha_t = \alpha(s_t, a_t, \mathbf{u_t}) = cp(s|\mathbf{u_t})/(1 + visits(s_t, a_t))$$

where c is a scaling constant and $p(s|\mathbf{u})$ is the probability density function of having the current discrete state given the current continuous state under the Gaussian distribution, $p(s_k|\mathbf{u_t}) = (2\pi\sigma^2)^{-1/2}\exp(-\|\mathbf{u}_t - \mathbf{z}_k\|_2^2/(2\sigma^2))$. This can be considered as a weight of the learning rate.

Algorithm

At step t,
Input: $s_t = s, s \in S_l$
$Q_{max} = \max_{a' \in A} Q(s, a')$ valid for only unique maximum value.
1. **if** a solution has not yet found **then**
2. $\quad Q_{ave,s} = (\sum_{a' \in A_l} Q(s, a'))/n$
3. $\quad$ **if** $visits(s, a) > 0$ for all $a \in A_l$ **or** $Q_{max} > Q_{ave} + Q_{\delta,2}$
4. $\quad\quad$ **if** $Q_{max} > (Q_{ave} + Q_{\delta,1})$**or** $\sum visits(s, \cdot) > 3|A_l|$ **then**
5. $\quad\quad\quad a_t = \text{argmax}_{a' \in A} Q(s, a')$
6. $\quad\quad\quad$ **if** $visits(s_t, a_t) > 4$ **then**
7. $\quad\quad\quad\quad \pi(s) = a_t$
8. $\quad\quad\quad$ **end if**
9. $\quad\quad$ **else**
10. $\quad\quad\quad a_t = \text{random}(a, a \in A_l)$
11. $\quad\quad$ **end if**
12. $\quad$ **else**
13. $\quad\quad a_t = a \quad$ where $visits(s, a) = 0, a \in A_l$
14. $\quad$ **end if**
15. **else**
16. $\quad a_t = \pi(s)$
17. **end if**
18. **Action Execution**
19. **Getting Rewards**: $r_t = \sum_{i=1}^{5} w_i(\tanh(c_i(x_{i,t} - \nu_i)) + b_i)$
20. **Q-update:**
$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \left[r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right]$

Table 1: Learning Algorithm

Result and Evaluation

Applying the RL model and the learning algorithm (table 1) in the simulation, we obtained the final Q values (Figure 1a) and the optimal policy. The figure 1b shows the map of the policy. The stand-up motion using the policy was compared with *HS* in cases of eight different initial falling down poses including the prone position (case 7) and the supine position (case8). The upper plot in the figure 1c shows the number of transitions and the lower plot shows its stability in each motion. We can see that the new stand-up motion is more stable in all cases except the case 8 with a little difference. In addition, the number of transitions of the new stand-up motion is less than or equal to the *HS* motion except the case 6 when a more stable set of motion sequences is required.

This approach can be expanded to more complicated environments such as inclined planes, uneven ground, or environments with obstacles by applying more advanced methods in RL.

References

George Konidaris, Sarah Osentoski, P. T. 2011. Value Function Approximation in Reinforcement Learning using the Fourier Basis. *Association for the Advancement of Artificial Intelligence*.

Jörg Stuückler, Johannes Schwenk, S. B. 2003. Getting Back on Two Feet: Reliable Standing-up Routines for a Humanoid Robot. *IOS Press*.