Learning Multi-Goal Inverse Kinematics in Humanoid Robot

Phaniteja Singamaneni*[1], Parijat Dewangan*[1], Abhishek Sarkar[1], and Madhava K. Krishna[1]
[1]Robotics Research Centre, International Institute of Information Technology, Hyderabad, India

Abstract

General Inverse Kinematic (IK) solvers may not guarantee real-time control of the end-effectors in external coordinates along with maintaining stability. This work addresses this problem by using Reinforcement Learning (RL) for learning an inverse kinematics solver for reachability tasks which ensures stability and self-collision avoidance while solving for end effectors. We propose an actor-critic based algorithm to learn joint space trajectories of stable configuration for solving inverse kinematics that can operate over continuous action spaces. Our approach is based on the idea of exploring the entire workspace and learning the best possible configurations. The proposed strategy was evaluated on the highly articulated upper body of a 27 degrees of freedom (DoF) humanoid for learning multi-goal reachability tasks of both hands along with maintaining stability in double support phase. We show that the trained model was able to solve inverse kinematics for both the hands, where the articulated torso contributed to both the tasks.

1 Introduction

The problem of inverse kinematics [1, 2] in any robotic system is defined as a mapping from the end-effector coordinates to actuator space i.e.,

$$\theta = f^{-1}(\mathbf{x}) \qquad (1)$$

where $\theta \in \mathbf{R^n}$ are the joint angles and $\mathbf{x} \in \mathbf{R^m}$ represents the position and orientation vector of the end effector. In complex and redundant robotic systems like humanoid robots, finding an inverse kinematic solution is not straightforward and many a times it is a highly ill-posed problem. This is because of the non-uniqueness of IK solution. Many of the existing solvers [3, 4, 5, 6] provide unique solutions without exploring the redundancy existing in the robots. This limits their application to humanoid robots where constraints like stability and self collision avoidance has be ensured along with the inverse kinematic solution. Several redundancy resolution methods [7, 8] had been proposed in the case of manipulators, but these methods become computationally expensive and may not yield a real time solution due to presence of large number of DoF in humanoid robots.

A much more complex problem to explore in this setting is to solve IK for multiple end effectors simultaneously. One direction would be using the above mentioned solvers in sequential manner at each step of IK. In order to enable simultaneous solving of multiple end effectors in the presence of constraints, augmented Jacobian [9] and extended Jacobian [10] were proposed. However these methods suffer from algorithmic singularities [11] apart from above mentioned limitations. Learning based methods [12] serve as a very good alternative that can overcome these limitations as well as solve the IK in real time satisfying the required constraints. Especially, reinforcement learning [13] provides a suitable framework for exploring the entire solution space and learning a generalized IK solver. The most important criteria for a humanoid robot is stability and hence IK solver should ensure that this is not violated in the final solution. Therefore, it is necessary to include a measure of stability as a constraint into the IK solver. Some of the recent advancements in RL like Deep Q-learning (DQN) [14], Guided Policy Search [15], Trust region policy optimization [16] and Deep Deterministic Policy Gradient (DDPG) [17] provide us many frameworks for learning a generalized IK solver in the presence of such constraints. Among these frameworks, DDPG was proposed in the context of learning continuous control tasks and hence it becomes a natural choice to solve the problem of IK. In this paper, we propose DDPG based IK solver for computing IK of both the hands of humanoid simultaneously. The proposed IK solver will also take into account, the criteria of stability and self-collision avoidance while

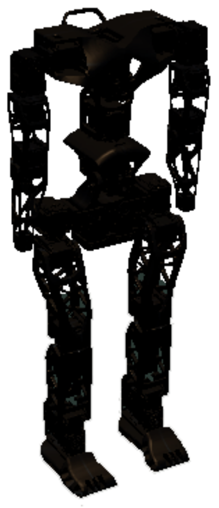


Figure 1 27 DoF humanoid robot with articulated torso.

*These authors have equal contribution.

generating joint space configurations. Zero Moment Point (ZMP) [18, 19] is used as the measure of stability and self-collision detection is performed using the bounding box approach [20].
The rest of the paper is organized as follows: Section 2 discusses the background needed for the framework. The proposed framework for multi-goal inverse kinematics is explained in section 3. Consequently, section 4 presents the results and simulations for the proposed framework. Finally conclusions are presented in section 5.

2 Background

Robot learning is a field which is at an intersection of machine learning and robotics. It allows a robot to acquire a skill or to adapt its environment through learning algorithms. Reinforcement learning is an area in machine learning where agents ought to take actions in an environment in order to maximise some notion of cumulative rewards. RL can be used to train an agent which learns from the environment directly without the use of any external data.
A recent work in deep RL for learning control tasks [21] learns the door opening tasks using asynchronous off-policy updates, combining the updates from multiple robots. In another recent work [22], a humanoid robot learns to perform a single-goal reachability task using RL. In our work, we concentrate on learning dual-goal reachability tasks simultaneously using both hands as end-effectors, unlike [22] where only one hand is reaching the goal. It should be noted that simultaneous dual goal reaching problem is very complex to tackle and IK is not straightforward. This work proposes an approach to learn IK for dual arm reachability tasks in a humanoid robot based on deep RL.
For learning an IK solver, we need to setup up the environment for training. For training, we consider a standard reinforcement learning setup with an agent interacting with the environment E in discrete time steps. At each time step t, the agent takes a state $s_t \in S$ as input and performs an action $a_t \in A$ according to the policy $\pi : S \rightarrow A$ and receives the observations. These observations consist of reward, denoted by r_t $(\in R)$ and the future state, denoted by s_t'. A fully observable Markov Decision Process [13] is assumed with state space S, action space A, a reward function $r(s_t, a_t)$ and the initial state, s_1 is assumed to be drawn from distribution $p(s_1)$. We also assume an infinite horizon with a discount factor γ. The goal in any reinforcement learning algorithms is to maximize the expected return:

$$R_t = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t),$$

where $\gamma \in [0, 1]$. The discounted state visitation distribution for a policy π is denoted by ρ^π. In this work, we consider only deterministic policies, i.e., $\pi(s_t) = a_t$.
Action-value function $Q(s_t, a_t)$ which is used in many RL algorithms is defined as the expected return after taking an action a_t in the state s_t and thereafter following the given policy π:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[R_t | s_t, a_t].$$

Thereby we can see that maximizing the action-value function results in maximization of the expected return. The expectation depends only on the environment and hence Q^π can be learnt off-policy, using transitions from a different behavioural policy β. Hence, the action value function can be written as:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s \sim \rho^\beta, a_t \sim \beta, r_t \sim E}[R_t | s_t, a_t]. \tag{2}$$

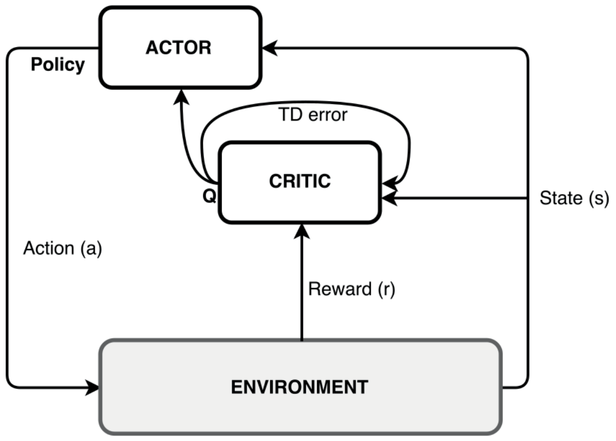


Figure 2 Actor Critic Algorithm. Actor (Policy) is updated based on the feedback from critic. Critic (Q) is updated using Temporal Difference (TD) [13] method.

An actor-critic based policy learning algorithm implements a generalized policy iteration that alternates between a policy evaluation and a policy improvement step. The critic evaluates the current policy and the actor tries to improve the current policy using feedback from the critic. This process is shown diagrammatically in Fig 2. In this paper we use DDPG, an actor-critic based algorithm that uses double Q learning [23] and Deterministic Policy gradient (DPG) [24] for policy evaluation and improvement respectively. The complete algorithm and its application for multi-IK is explained in the following section.

3 Learning Multi-goal Inverse Kinematics for Humanoids

Motion planning for humanoids is a daunting task given that they typically have 25 or more DoF. The problem is further complicated by the fact that humanoids must be controlled carefully in order to maintain dynamic stability. Thus, existing motion planning techniques may not be applicable directly on humanoid robots. Hence, there is need for efficient IK solvers for humanoids that can be easily employed into motion planning algorithms. In this section, we explain the proposed methodology and the underlying modelling to learn inverse kinematics for multi goal reachability tasks in humanoid robots.

3.1 Robot Model

The robot model used for study is shown in Fig 1. The robot has a total height of 84cm, total weight below 5Kg and 27 DoF. The main feature of this design is the 5 DoF articulated torso, which enhances its flexibility and stability. We can notice that the upper body constitutes of two closed kinematic chains, where the torso is the common sub-chain. In order to learn inverse kinematics of the 13 DoF upper body, we need to model the common chain (articulated torso) in a effective way. In the following subsection, the procedure to learn a generalized IK solver for this system is explained.

3.2 Multi IK using DDPG

DDPG is a deep reinforcement learning algorithm that deals with continuous control tasks. It is an actor-critic based algorithm that continuously improves the policy as it explores the environment as explained in Section 2. Both actor and critic are approximated using neural networks (θ^{μ}, θ^{Q}). This algorithm uses off-policy mini-batch update using replay buffer in order to ensure that data is sampled from independently and identically distributions. The problem of instability in training is addressed by using target networks ($\theta^{\mu'}$, $\theta^{Q'}$).

Suppose $Q(s,a|\theta^{Q})$, $\mu(s|\theta^{\mu})$ represent critic and actor networks respectively and $Q'(s,a|\theta^{Q'})$, $\mu'(s|\theta^{\mu'})$ represent their corresponding target networks. Given this parametrization, the critic network weights are optimized by minimizing the following loss function:

$$L(\theta^{Q}) = (Q(s_t,a_t|\theta^{Q}) - y_t)^2 \tag{3}$$

where,

$$y_t = r(s_t,a_t) + \gamma Q'(s_{t+1}, {\mu'}_{t+1}(s_{t+1}|\theta^{\mu'})|\theta^{Q'}). \tag{4}$$

According to [24], the deterministic policy gradient is given by:

$$\nabla_{\theta^{\mu}} J = \nabla_a Q(s_t,a_t|\theta^{Q}) \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu}). \tag{5}$$

Suppose the learning rate is η, then the update on actor network is given by:

$$\theta^{\mu} = \theta^{\mu} + \eta \nabla_a Q(s_t,a_t|\theta^{Q})|_{a=\mu(s_t|\theta^{\mu})} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu}). \tag{6}$$

The target networks follow slow updates according to Eq. (7), where $\tau << 1$.

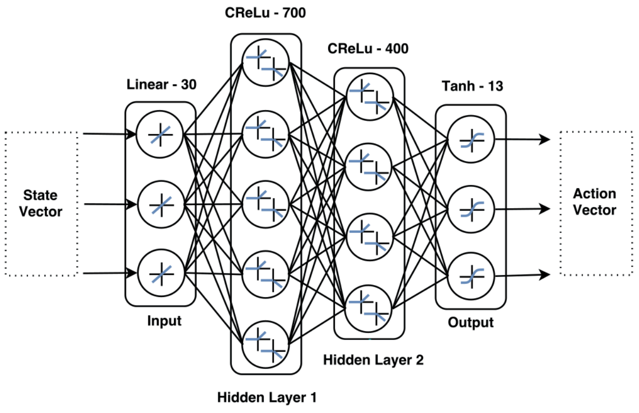
$$\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{aligned} \tag{7}$$

3.2.1 State vector and network architecture

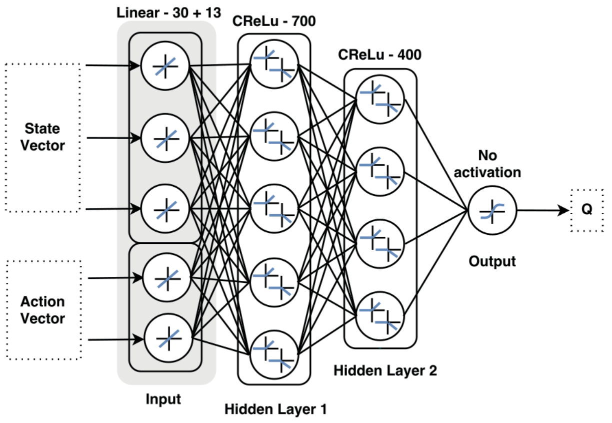
The chosen state vector consists of all the joint angles (**q**) of the upper body, end-effectors coordinates and both goal coordinates. It also contains flags related to collision, stability and coordinates of last joint of spine where both arms are connected. The actor network is fed in with state vector and it outputs the action vector that contains angular velocities of all the joints needed to move towards the goals. The critic network takes state-action vector as input and outputs its corresponding action-value. The state and action vector dimensions used in our model are 30 and 13 respectively.

Both actor and critic networks are fully connected two hidden layers each. In both networks, the first hidden layer consists of 700 units and the second hidden layer consists of 400 units with *CReLu* activation in both the layers. Batch normalization and a drop-out of 20% is present in both hidden layers of actor. Tanh activation is used in the output layer of actor. In critic network, first hidden layer has a drop-out of 30% and batch normalization, whereas the second hidden layer has a drop-out of 20% with no batch normalization. L2 regularization of 0.01 is used in all layers of the critic network. The output layer in critic don't have any activation function.

The network architecture for the actor and critic is given in Fig 3. Note that use of *CReLu* doubles the depth of the activations.



(a) Actor network : Takes state vector as input and outputs action vector. The activation function and the number of units are shown on the top of each layer.



(b) Critic network: Takes concatenated state-action vector as input and outputs action value function, Q. The activation function and the number of units are shown on the top of each layer.

Figure 3 Actor and Critic Network Architectures

3.2.2 Reward function

In deep RL frameworks, reward function is an integral part of the network update and hence the underlying policy that is learnt by the network. For learning multi-goal inverse kinematics, we need to devise a global reward function such that it addresses reachability tasks of both the hands simultaneously. The main objective of an IK problem is to provide a set of angles ($\mathbf{q}$) that are needed to reach the given goal position.

In order to ensure that the configurations given out by the solver are within the stability region, a large negative reward is given whenever it goes out of stability bounds. The final reward function is shown below.

$$r_i = \begin{cases} -\alpha dist_i & if\, stable\, and\, collision free \\ -\kappa & if\, unstable\, or\, collides \end{cases} \tag{8}$$

$$if\, hand_i\, reaches\, goal\, then\ r_i = r_i + \lambda \tag{9}$$

If both the hands reach the goal simultaneously then

$$r_i = \beta \qquad \forall i = 1,2$$

where r_i denotes reward of i_{th} task. The global reward returned by the environment is given by the sum of rewards of both the tasks, that is,

$$r = r_1 + r_2$$

where α, β, κ are the normalization constants, $dist_i$ is the absolute distance between goal position and end effector position for hand i where $i = 1,2$.

3.2.3 Environment setting and Training

Algorithm 1 Multi-IK learning using DDPG

```
1: Randomly initialize actor-critic networks. Copy the
   weights into the target networks.
2: Initialize replay buffer R.
3: for $i = 1$ to $MaxEpisodes$ do
4:     $s$ = env.Reset()
5:     for $j = 1$ to $MaxStep$ do
6:         $a$ = actor.get_action($s$) + $N$        ▷ $N$ is exploration noise
7:         $s', r, tr$ = env.Step($a$)
8:         R.store($s, a, r, s'$)
9:         if (R.size() > $batch\_size$) then
10:            $batch$ = R.sample($batch\_size$)
11:            critic.update($batch$)
12:            actor.update(critic.gradients,$batch$)
13:            actor.target_update(), critic.target_update()
14:        end if
15:        if ($done$ or $tr$) then
16:            $break$
17:        end if
18:    end for
19: end for
```

In order to learn a generalized inverse kinematics, the entire workspace need to be spanned. To ensure that, the start configuration and goal positions are sampled randomly at the start of each episode. This is done in the env.Reset() step shown in Algorithm 1. In env.Step(a), robot environment executes the action a and returns next state s', reward r and terminate flag tr. The terminate flag will be one if there is self-collision or the robot loses balance.

The robot is trained in MATLAB using Robotics toolbox and tested in MuJoCo, a dynamic simulation environment. The training setting is modelled in TensorFlow code-base in Python. RMSProp optimizer is used to train both actor and critic networks with learning rate 0.0001 for both. Target networks are used for both critic and actor networks. These target networks are copied into their corresponding actor and critic network after every 1000 and 1100 steps respectively. We used a replay buffer (R) of size 50000 to store the information of each step. Training was run for 10000 episodes with 150 steps in each episode. A mini-batch of size 64 is sampled randomly from the replay buffer for training the networks. A normally distributed decaying noise function was used for exploration. The episode was terminated in between if the robot went unstable or self-collided. Complete algorithm for training is shown in algorithm 1. Results and observations of training are presented and discussed in the subsequent section.

4 Results and Simulations

The humanoid robot shown in Fig. 1 was trained for reachability tasks of both the hands, taking into account stability and collision avoidance using the reward setting explained in previous section. For incorporating stability criteria in the reward function, we used Zero Moment Point[25], a dynamic stability check measure. It is calculated as:

$$\begin{aligned} p_x &= \frac{Mgx - \dot{L}_y}{Mg + \dot{P}_z} \\ p_y &= \frac{Mgy + \dot{L}_x}{Mg + \dot{P}_z} \end{aligned} \tag{10}$$

where x, y are the x and y coordinates of center of mass, M is the total Mass of the robot, g is acceleration due to gravity and [L_x, L_y, L_z], [P_x, P_y, P_z] are the angular and linear momentum respectively with respect to base frame.

4.1 Training Results

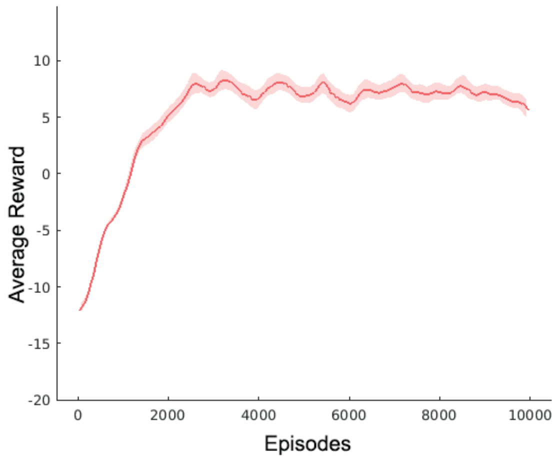


Figure 4 Average reward: X-axis shows episodes and Y-axis shows the value of the reward. Coloured region is 95% confidence interval.

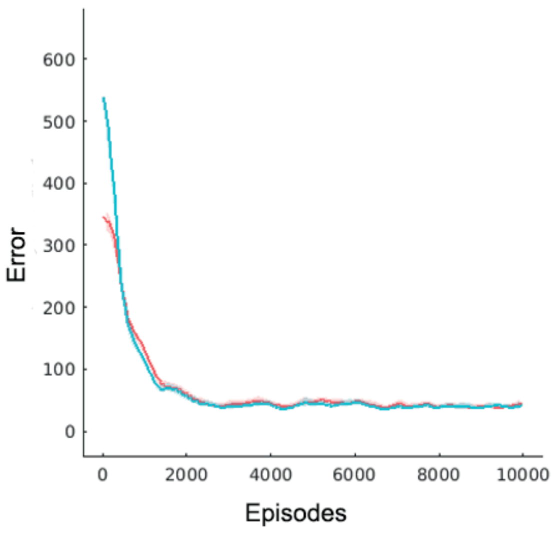


Figure 5 Average error graphs of left (red line) and right (blue line) hand. X-axis shows episodes and Y-axis show value of the errors.

In order to evaluate the learning process, average reward per episode and error per episode are plotted, which are shown in Figs. 4,5. Fig. 4 shows the average reward curve with episodes during training. The bold line shows the reward value and coloured region around shows the 95% confidence interval. It can be observed that the average reward increases gradually and saturates showing completion of the training process.
In Fig. 5, the error curves of both the hands from their respective goals at the end of each episode are plotted. It can observed that these errors decrease gradually and saturates to a value very close to zero. This signifies the completion of the task, as error depicts the distance between the hand and their respective goals. In Fig. 5, each coloured line indicates the error corresponding to one task. As both the tasks were trained simultaneously, we can see that the both the curves of average error have similar profile.

It can be observed that both reward and error plots started to saturate after 3000 episodes which shows that the optimal solution has been attained, therefore indicating that the network has learnt the optimal IK solver.

4.2 Testing in dynamic simulator

The learnt IK solver is tested by providing several random goal positions. For testing, we used a dynamic simulator, MuJoCo, instead of MATLAB, to show the transferability of the solver to a more realistic environment. Some snippets of humanoid from this testing are shown in Fig. 6. The snippets show two different goal position settings for both the hands, and it can be seen that the solver was able to successfully solve in both settings. We can also observe that the start, intermediate and the final positions are stable configurations. This shows the framework was able to learn the final IK solution as well as the joint trajectories needed to reach given goal positions. Thus,

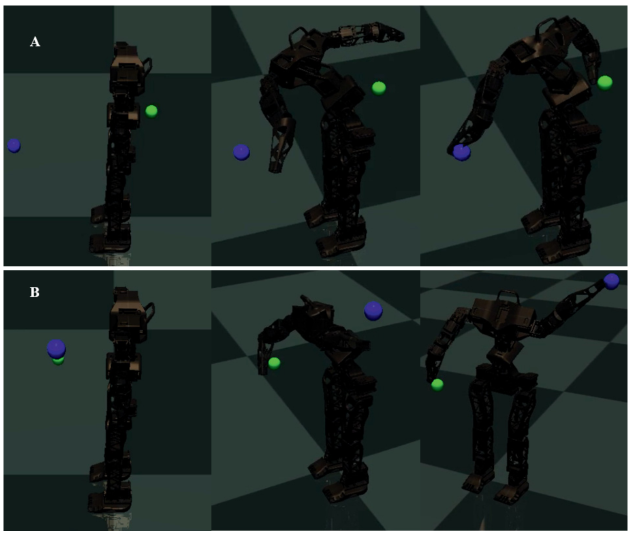


Figure 6 Snippets of two cases of humanoid simulation in MuJoCo, where the trained model is tested for multi-goal reachability task. In A, one goal position is the front and another in back. In B, both goal positions are in front but one is upwards whereas the other one is just in front of pelvis.

we can conclude that the framework has learnt a stable multi-goal IK solver for 27 DoF humanoid which gives stable configurations at all intermediate steps.

The above learnt IK solver can be easily transferred to a real robot by using velocity level controllers. This can be done easily by providing the goal positions to the IK solver and then using the solver in the feedback loop of the velocity controllers.

5 Conclusion

In this paper, a methodology to learn inverse kinematics of multiple open kinematic chains with shared sub-chains was proposed. The proposed methodology was based on DDPG and was able to learn IK solver that can simultaneously solve IK for multiple end effectors. A vivid description of the method and the networks used are presented along with reward function modelling.
The proposed framework was applied to learn generalized IK solver for a 27 DoF humanoid with 5 DoF articulated torso for reachability tasks of both hands and results were presented. Results show that the framework is capable of learning inverse kinematics (also joint trajectories), along with maintaining stability and self-collision avoidance. Although the proposed model has limitations like collision avoidance and accuracy, this model can be good prototype for solving inverse kinematics on highly redundant manipulators.

Future work involves making the framework more versatile such that it could be extended to various other agents as well. We will also focus on including collision avoidance with the environments into the framework.

6 Literature

[1] A. Colomé, "Smooth inverse kinematics algorithms for serial redundant robots," Ph.D. dissertation, Master Thesis, Institut de Robotica i Informatica Industrial (IRI), Universitat Politecnica de Catalunya (UPC), Barcelona, Spain, 2011.

[2] Y. Chua, K. P. Tee, and R. Yan, "Robust optimal inverse kinematics with self-collision avoidance for a humanoid robot," in *RO-MAN, 2013 IEEE*. IEEE, 2013, pp. 496–502.

[3] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on man-machine systems*, vol. 10, no. 2, pp. 47–53, 1969.

[4] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[5] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[6] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural networks*, vol. 6, no. 4, pp. 525–533, 1993.

[7] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[8] J. Hollerbach and K. Suh, "Redundancy resolution of manipulators through torque optimization," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 308–316, 1987.

[9] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of intelligent and robotic systems*, vol. 3, no. 3, pp. 201–212, 1990.

[10] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2. IEEE, 1985, pp. 722–728.

[11] J.Baillieul, "A constraint oriented approach to inverse problems for kinematically redundant manipulators," in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4. IEEE, 1987, pp. 1827–1833.

[12] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2001, pp. 298–303.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[14] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *AAAI*, 2016, pp. 2094–2100.

[15] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1–9.

[16] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1889–1897.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[18] M. Vukobratović and J. Stepanenko, "On the stability of anthropomorphic systems," *Mathematical biosciences*, vol. 15, no. 1-2, pp. 1–37, 1972.

[19] M. Dekker, "Zero-moment point method for stable biped walking," *Eindhoven University of Technology*, 2009.

[20] C. Dube, M. Tsoeu, and J. Tapson, "A model of the humanoid body for self collision detection based on elliptical capsules," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2397–2402.

[21] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3389–3396.

[22] S. Phaniteja, P. Dewangan, P. Guhan, A. Sarkar, and K. M. Krishna, "A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2017, pp. 1818–1823.

[23] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, 2010, pp. 2613–2621.

[24] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 387–395.

[25] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to humanoid robotics*. Springer, 2014, vol. 101.