


RESEARCH ARTICLE

Motion recognition using deep convolutional neural network for Kinect-based NAO teleoperation

Archana Balmik* , Arnab Paikaray, Mrityunjay Jha and Anup Nandy

Department of Computer Science and Engineering, NIT Rourkela, Rourkela, India

*Corresponding author. E-mail: archanabalmik1392@gmail.com

Received: 28 July 2021; **Revised:** 8 December 2021; **Accepted:** 20 January 2022; **First published online:** 28 February 2022

Keywords: teleoperation, humanoid robot NAO, Kinect v2, human motion recognition, balance control strategy, convolutional neural network

Abstract

The capabilities of teleoperated robots can be enhanced with the ability to recognise and reproduce human-like behaviour. The proposed framework presents motion recognition for a Kinect-based NAO teleoperation. It allows the NAO robot to recognise the human motions and act as a human motion imitator. A stable whole-body imitation is still a challenging issue because of the difficulty in dynamic balancing of centre of mass (CoM). In this paper, a novel adaptive balancing technique for NAO (ABTN) is proposed to control the whole body in single as well as double supporting phases. It targets dynamic balancing of the humanoid robot by solving forward kinematics and applying a weighted average of mass with the CoMs of individual links with respect to the previous joint frames, which provides us with the dynamic CoM of the whole body. Our novel approach uses this dynamic CoM and calculates joint angles using proposed pitch and roll control algorithm to keep the dynamic CoM inside the stable region. Additionally, the NAO robot is capable of recognising human motions using the proposed 7-layer one-dimensional convolutional neural network (1D-CNN). To solve the problem of variable length of time sequences, Zero padding is introduced with 1D-CNN. It attains a recognition accuracy of 95% as compared to the hidden Markov model and neural network. The experimental results demonstrate that the developed teleoperation framework is robust and serves as potential support for the development and application of teleoperated robots.

1. Introduction

Teleoperation has become a popular and emerging trend in research and development. It arises from the combination of various fields of research such as computer vision, communication & control, artificial intelligence, mechatronics, machine learning, deep learning. [1]. The teleoperation bridges the gap between humans and the robots. The controlling of robotic operations from a distance is termed as teleoperation [2]. The distance varies from billions of kilometres as in space applications to centimetres as in micro-applications or microsurgery. It extends the human capability to the places where it is impossible to present physically, dangerous or highly threatening to human life, for example disaster scenarios, radioactive environment, chemical industries, deep oceans, minefields. [3]. The replacement of individual with a robot that can be controlled remotely is the most desirable solution to these scenarios [1]. Recent studies reported that humanoid robots have made substantial progress in different application of teleoperation [4, 5, 6, 7, 8, 9]. Generally, humanoid robots are designed to perform specific tasks in teleoperation. But it is quite difficult to control the humanoid robots with the normal controllers due to its high degree of freedom [10]. Therefore, a suitable interface is needed for controlling the robots. It has a significant influence on effective human-robot interactions (HRI) in teleoperation. The most common interfaces that have been used in HRI are keyboard, mouse, dials, joystick, gloves, inertial sensors, and exoskeletal systems [11]. These are mechanical contacting devices which require unnatural arm or hand motions to accomplish tasks in teleoperation. As a consequence, the new interfaces for HRI have

been developed to provide more comfortable ways of interacting with remote robots. Many researchers have integrated sensors technique interfaces such as Microsoft Kinect [8, 9, 12–14]. The Kinect sensor provides contactless, user-friendly, more human-centred interface for teleoperation. Moreover, it encourages the non-experts to interact with robots. Additionally, it is easy-to-use and has advantages of low cost [15]. The 3D Kinect camera simplifies the human-robot interaction process by capturing different human motions to control the remote robots. It allows to develop more natural HRI systems by identifying different motions of the human operator.

The similarities between humanoid robots and humans have increased the cooperation between humans and robots [16]. Therefore, teleoperation research studies focus on human behaviour imitation and recognition to establish natural communication with robots. Many researchers recently developed imitation-based teleoperated systems that capture human motions and map them to robots [10, 17, 18, 19]. The motivation behind these systems is to integrate human motions for interacting with robots same as human use motions for pointing to objects or while speaking. It has potential to control the robots in complicated situations where it is too difficult to programme a robot [10]. In the last few decades, human motion/gesture recognition is gaining a lot of attention from robotics groups. The increasing interest is due to the affordability of depth sensors such as Kinect. It provides RGB data along with the depth information; hence, it is called as RGB-D sensors [20]. Several researchers presented human motion recognition in their literatures which used Kinect sensors [21, 22, 23, 24, 25]. These studies either used depth images [26, 27, 28] or skeleton data [3, 29, 30] to classify the arbitrary motions. It serves reliable teleoperated systems where a robot predicts the human behaviour towards them to perform the associated tasks.

The contribution of this work consists of three aspects:

1. A Kinect-based teleoperation for NAO robot is presented in which the robot imitates the full-body human motions.
2. A novel Adaptive balance technique for NAO (ABTN) is proposed to balance the whole body.
3. A deep 1D-CNN is developed for the robot to recognise the human motions with a high accuracy.

The remaining paper is arranged as follows: A comprehensive literature study on teleoperation and convolution neural network-based human motion recognition is presented in Section 2. The detailed description of the proposed framework is given in Section 3. The experimental results and comparison with the state-of-the-art methods are examined in Section 4. And, Section 5 concludes the paper and discusses the future work.

2. Related work

The previous studies related to gesture recognition and imitation of human motions in order to control the robots are presented in this section. Vongchumyen et al. [7] presented a teleoperation of a humanoid robot that uses the depth sensor of the Kinect to capture the human motions. The joint positions are considered as vector to find out the joint angles. It controls the servo motors of the robot using Restful API. The developed system is able to control only the upper body of the robot. Zhang et al. [31] proposed an algorithm for NAO robot to mimic the human whole-body motions. It uses eight kinematic chains to map human motions to the robot for imitating the actions. Stable motions are achieved by constraining the CoM. Chen et al. [8] performed the imitation of whole-body human motions for NAO. It introduces new computing methods for joint angles to control the robot. After analysing the differences in the joint angles, a gain factor is defined for mapping of joint angles during the control process. It also slows down the changing rate of joint angles to stabilise the robot motions. Alquisiris et al. [1] developed a system to control the NAO robot through gestural interpretation. The Kinect sensor is used to capture the user's movements. The joint angles are obtained using linear regression methods and achieve whole-body imitation of human motions. But in this system, if the motions are not smoother enough, then the

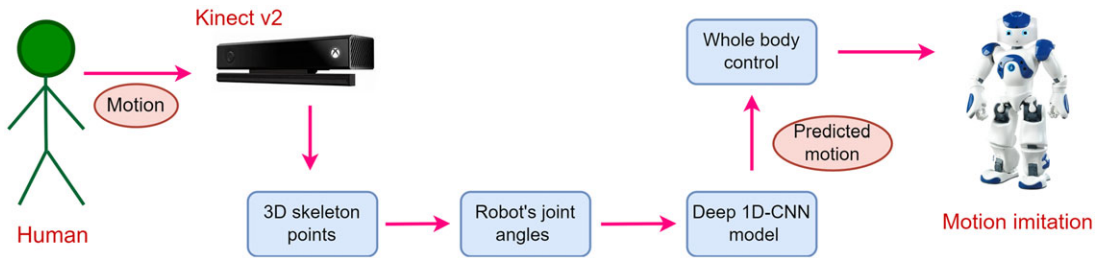


Figure 1. Overview of the proposed framework.

robot abruptly falls. The balance control of the lower body is necessary for the robot performing various tasks in teleoperation. The problem of maintaining whole-body balance is quite critical. But in previous studies, fewer methods are introduced for balance control. Either the changing rate of joint angles is slowed down or smoother motions are taken. Here, we proposed a novel ABTN method to solve this problem. This method determines the dynamic centre of mass (CoM) and computes the joint angles of the ankle from knee and hip joints to avoid falling off the robot while performing any whole-body motions.

Several gesture recognition approaches such as hidden Markov model (HMM) [29], support vector machine (SVM) [32], extreme learning machines (ELM) [16], artificial neural network (ANN) [33], convolutional neural network (CNN) [34], dynamic time wrapping [32] have been applied to recognise the postures in teleoperation. In the last couple of years, literature has proved that one dimensional CNN is the most effective approach for human motion recognition [35, 36, 37, 30]. Hu and Xu [37] proposed CNN for skeleton-based human action recognition. It uses two convolutional layers in the architecture for learning the spatio-temporal features. Data augmentation and segment pooling are implemented for long sequences recognition. Li et al. [36] employed CNN framework for human action detection and classification. The skeleton coordinates are fed directly to a 7-layer CNN model. It achieves 89.3 accuracy for the NTU RGB+D dataset. Kiranyaz et al. [38] reported that 1D-CNN has low computational complexity and is easier to train than 2D or 3D CNNs. It makes the real-time and low-cost hardware implementation feasible because of its simple and compact architecture. Despite having several advantages, there are very few literatures on 1D-CNN focusing on skeletal motion recognition, which remains this field unexplored. Therefore, one-dimensional convolutional neural network (1DCNN) is proposed to identify the human motions using skeletal data and the model performance is compared with the state-of-the-art methods to achieve remarkable results.

3. Proposed framework

In the proposed system framework, a motion capturing device that is Kinect v2 is used to capture the human motions. The 3D skeleton positions of motions are obtained from the motion capturing device. The captured motion information has some breakpoints in the times series. Therefore, human skeleton positions are interpolated using cubic spline interpolation. It samples the data uniformly and provides a smoother joint motion. The interpolated human joint positions are used to extract the joint angles as a feature for this experimental study. An analytical method presented by Zhang et al. [39] is used for mapping the human motions to the humanoid robot NAO. The extracted joint angles are filtered using Savitzky-Golay filter of 5th order with a window size of 13. It smoothens the joint motions by removing noise present in the data. The filtered joint angles are given to the CNN module, which identifies the human motions to be performed by the robot. The framework employs a whole-body control mechanism so that the humanoid robot does not fall while performing the imitation of recognised motion. The overview of the proposed framework is presented in Fig. 1. The proposed 1D-CNN architecture along with the whole-body control mechanism for the humanoid robot is described in detailed in below subsections.

3.1 Centre of Mass (CoM)

The estimation of the CoM is vitally important for whole-body balance when a slight change in the CoM may lead to catastrophic effects. For calculating the CoM of humanoid robot NAO, we have used forward kinematics to traverse through the kinematic chain and then multiply the local CoM with respect to the link. The weighted average of all the CoMs will provide the CoM of the whole body.

Affine Transformations: Affine transformation is a linear mapping method that transforms vectors from one coordinate space to another preserving dimensional ratios. An affine transformation matrix is a $(n + 1) * (n + 1)$ matrix where n is the number of dimensions [40]. Here, we have used translation and rotation affine transformation.

Translation Matrix: The translation matrix moves vectors by a fixed distance in a specific direction. Here dx , dy , and dz define the distance of translation along the x , y , and z -axis, respectively.

$$A(dx, dy, dz) = \begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation Matrix: The rotation matrix rotates vectors by a fixed angle about a specific direction. Here, R_x , R_y , and R_z are rotation matrices that rotate the vectors about the x , y , and z -axis, respectively, by an angle θ .

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) & 0 \\ 0 & \sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_y = \begin{pmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Denavit and Hartenberg Parameters: The Denavit and Hartenberg (DH) transformation is used to define the motion of actuators connected by rigid links of a kinematic chain. Firstly, the Z -axis is defined along the axis of rotation/translation of the parent joint. The DH parameters are derived from common normal between consecutive Z -axes.

- d – depth along the previous joint’s z -axis
- θ – angle about the previous z -axis to align its x with the new origin
- a – distance along the rotated x -axis
- α – rotates about the new x -axis to put z in its desired orientation.

$$T(a, \alpha, d, \theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & a \\ \sin(\theta) \cos(\alpha) & \cos(\theta) \cos(\alpha) & -\sin(\alpha) & -d \sin(\alpha) \\ \sin(\theta) \sin(\alpha) & \cos(\theta) \sin(\alpha) & \cos(\alpha) & d \cos(\alpha) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Whole-body CoM: The whole-body CoM is the weighted average of the individual joint CoM matrices. The CoM of individual kinematic chain $C(I,N)$ is calculated as the cross product of affine matrices of joints and translation matrices of CoM with the mass of individual links.

TotalCoM = 1/M * [TorsoCoM + HeadCoM + LArmCoM + RArmCoM + LLegCoM + RLegCoM]

$$C(I, N) = \sum_{i=I}^N \frac{m_i}{1000} J(I, i) * A(x_i, y_i, z_i) \tag{1}$$

where $M = M$ is the mass of the robot = 5.305350006 Kg

m_i = mass of i th link

J_i = affine transform matrix of i th joint with respect to torso frame (dimensions are in mm, thus divided by 1000)

$A(x, y, z)$ = translation matrix for CoM position with respect to the preceding joint

I = initial row of Tables I and II

N = final row of Tables I and II

$$J(i, n) = \prod_{j=1}^n A(dx_j, dy_j, dz_j) * T(a_j, \alpha_j, d_j, \theta_j) \tag{2}$$

where $A(dx, dy, dz)$ = translation matrix of joint offsets for specific kinematic chain

$T(a, \alpha, d, \theta)$ = DH transformation matrix for joints

The steps to calculate the whole-body CoM can be summarised as given below:

1. To calculate CoM of individual kinematic chain, we are considering corresponding initial to final rows of Table I, that is translation and DH parameters for different joints and Table II, that is masses and CoM parameters.
2. We then take the cross product of the Affine Matrix from Forward Kinematics in equation (2) and Translation Matrix of the CoM of individual links from Table II in equation (1).
3. Next, the Forward Kinematics in equation (2) of the respective kinematic chain is the cross product of the Translation Matrix of Joint Offsets [41] and DH Matrix of the Joint angles taken from Table I.

For example, for calculating torso CoM we are using $C(1, 1)$ in equation (1) by considering row 1 from both Tables I and II simultaneously where we multiply mass and COM of torso from Table II and determine kinematic chain of torso using DH parameters from Table I. Similarly, we have calculated HeadCoM = $C(2, 3)$ by considering rows 2 to 3 from both Tables I and II. Likewise for RightArmCoM = $C(4, 8)$, LeftArmCoM = $C(9, 13)$, RightLegCoM = $C(14, 19)$, LeftLegCoM = $C(20, 25)$.

3.2 Whole-body control

This module enables whole-body control using the proposed adaptive balancing technique for NAO (ABTN). This technique involves a novel mechanism for pitch and roll control inspired by the work presented in ref. [8]. Here, the joint angles of the ankle are computed from knee pitch and hip pitch to make the robot stand upright. The aim of this approach is to keep the dynamic CoM just calculated above inside the base area of its foot while performing whole-body motion. This is also robust for complex motions like one-leg stand where the total base area reduces to only one foot. It detects if the robot is about to perform one-leg stand and controls the ankle roll to balance the robot. This approach is experimented using Webots simulation software, ROS environment, and the Naoqi package. The computation procedure of joint angles for pitch and roll control to maintain the balance of NAO lower body are derived below

3.2.1 Pitch control

The pitch control is essential for single leg stand, crouching, and squatting postures. The proposed algorithm uses adaptive pitch control for ankle, knee, and hip joints to keep the CoM of the robot inside the foot base area.

Table I. Translation and DH translation parameters for joints of NAO robot.

j	Joints	Translation matrix (A) parameters			DH transformation (T) parameters			
		dx (mm)	dy (mm)	dz (mm)	a (mm)	α (rad)	d (mm)	θ (rad)
1	Torso	0	0	0	0	0	0	0
2	HeadYaw	0	0	NeckOffsetZ	0	0	0	head_yaw
3	HeadPitch	0	0	0	0	$-\pi/2$	0	head_pitch - $\pi/2$
4	RShoulderPitch	0	-ShoulderOffsetY	ShoulderOffsetZ	0	$-\pi/2$	0	r_shoulder_pitch
5	RShoulderRoll	0	0	0	0	$\pi/2$	0	r_shoulder_roll + $\pi/2$
6	RElbowYaw	0	0	0	-ElbowOffsetY	$\pi/2$	UpperArmLength	r_elbow_yaw
7	RElbowRoll	0	0	0	0	$-\pi/2$	0	r_elbow_roll
8	RWristYaw	0	0	0	0	$\pi/2$	LowerArmLength	r_wrist_yaw
9	LShoulderPitch	0	ShoulderOffsetY	ShoulderOffsetZ	0	$-\pi/2$	0	l_shoulder_pitch
10	LShoulderRoll	0	0	0	0	$\pi/2$	0	l_shoulder_roll + $\pi/2$
11	LElbowYaw	0	0	0	ElbowOffsetY	$\pi/2$	UpperArmLength	l_elbow_yaw
12	LElbowRoll	0	0	0	0	$-\pi/2$	0	l_elbow_roll
13	LWristYaw	0	0	0	0	$\pi/2$	LowerArmLength	l_wrist_yaw
14	RHipYawPitch	0	-HipOffsetY	-HipOffsetZ	0	$-\pi/4$	0	r_hip_yaw_pitch - $\pi/2$
15	RHipRoll	0	0	0	0	$-\pi/2$	0	r_hip_roll - $\pi/4$
16	RHipPitch	0	0	0	0	$\pi/2$	0	r_hip_pitch
17	RKneePitch	0	0	0	-ThighLength	0	0	r_knee_pitch
18	RAnklePitch	0	0	0	-TibiaLength	0	0	r_ankle_pitch
19	RAnkleRoll	0	0	0	0	$-\pi/2$	0	r_ankle_roll
20	LHipYawPitch	0	HipOffsetY	-HipOffsetZ	0	$-3\pi/4$	0	l_hip_yaw_pitch - $\pi/2$
21	LHipRoll	0	0	0	0	$-\pi/2$	0	l_hip_roll + $\pi/4$
22	LHipPitch	0	0	0	0	$\pi/2$	0	l_hip_pitch
23	LKneePitch	0	0	0	-ThighLength	0	0	l_knee_pitch
24	LAnklePitch	0	0	0	-TibiaLength	0	0	l_ankle_pitch
25	LAnkleRoll	0	0	0	0	$-\pi/2$	0	l_ankle_roll

Table II. Masses and CoM parameters of NAO robot [42].

i	Link name	Mass (Kg)	X (m)	Y (m)	Z (m)
1	Torso	1.0496	-0.00413	0	0.04342
2	Neck	0.07842	-1e - 05	0	-0.02742
3	Head	0.60533	-0.00112	0	0.05258
4	Right Shoulder	0.09304	-0.00165	0.02663	0.00014
5	Right Biceps	0.15777	0.02455	-0.00563	0.0033
6	Right Elbow	0.06483	-0.02744	0	-0.00014
7	Right ForeArm	0.07761	0.02556	-0.00281	0.00076
8	Right Hand	0.18533	0.3434	0.00088	0.00308
9	Left Shoulder	0.09304	-0.00165	-0.02663	0.00014
10	Left Biceps	0.15777	0.02455	0.00563	0.0033
11	Left Elbow	0.06483	-0.02744	0	-0.00014
12	Left ForeArm	0.7761	0.02556	0.00281	0.00076
13	Left Hand	0.18533	0.03434	-0.00088	0.00308
14	Right Pelvis	0.06981	-0.00781	0.01114	0.02661
15	Right Hip	0.14053	-0.01549	-0.00029	-0.00515
16	Right Thigh	0.38968	0.00138	-0.00221	-0.05373
17	Right Tibia	0.30142	0.00453	-0.00225	-0.04936
18	Right Ankle	0.13416	0.00045	-0.00029	0.00685
19	Right Foot	0.17184	0.02542	-0.0033	-0.03239
20	Left Pelvis	0.06981	-0.00781	-0.01114	0.02661
21	Left Hip	0.14053	-0.01549	0.00029	-0.00515
22	Left Thigh	0.38968	0.00138	0.00221	-0.05373
23	Left Tibia	0.30142	0.00453	0.00225	-0.04936
24	Left Ankle	0.13416	0.00045	0.00029	0.00685
25	Left Foot	0.17184	0.02542	0.0033	-0.03239

Ankle Pitch: For the calculation of ankle pitch, we have used the filtered nao robot joint angles and distance between the joints calculated from TF (transforms) or nao documentation. Figure 2 represents the side view of right leg joints of NAO robot in squat position where A, K, H, T are ankle, knee, hip, torso joints, and α_p, k_p, h_p, t_p are the respective joint angles. The vectors $\vec{AK}, \vec{KH}, \vec{HT}, \vec{TCoM}$ are used to represent the joint positions with base_footprint that is the point on the ground between two legs as the origin. From the geometry of the robot as in Fig. 2, the vector \vec{ACoM} connecting ankle joint and CoM is written as

$$\vec{ACoM} = \vec{AK} + \vec{KH} + \vec{HT} + \vec{TCoM} \tag{3}$$

The CoM should be inside the foot base area for which the real part of the \vec{ACoM} should be in the range $[-3, 9]$ cm. To ease the calculation part,

$$Real(\vec{ACoM}) = 0 \tag{4}$$

Let L_1 be the distance between ankle (A) & knee (K), L_2 be the distance between knee (K) & hip (H), L_3 be the distance between hip (H) & torso (T), and L_4 be the distance between torso (T) & CoM. Then, from Fig. 2,

$$L_4 = \sqrt{x^2 + z^2} \tag{5}$$

$$t_p = \tan^{-1} \left(\frac{x}{z} \right) \tag{6}$$

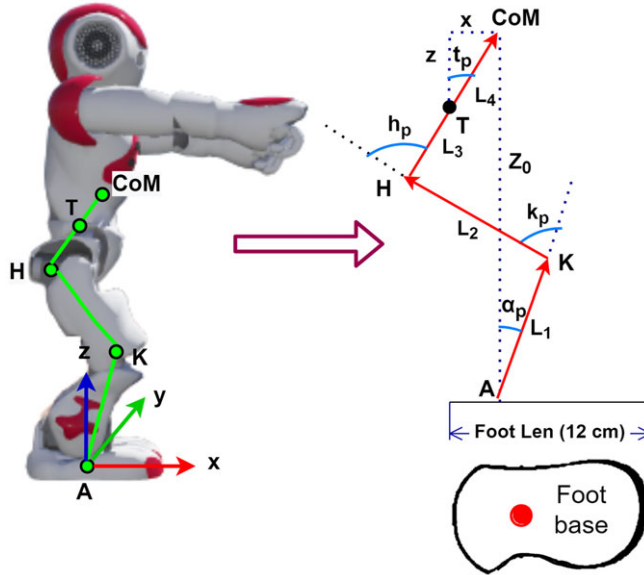


Figure 2. A side view representation of right leg joints of NAO robot in a squat position for pitch control.

$$\begin{aligned} &\Rightarrow L_1 \cos(90 + \alpha_p) + L_2 \cos(90 + \alpha_p + k_p) + L_3 \cos(90 + \alpha_p + k_p + h_p) \\ &\quad + L_4 \cos(90 + \alpha_p + k_p + h_p + t_p) = 0 \\ &\Rightarrow L_1 \sin(\alpha_p) + L_2 \sin(\alpha_p + k_p) + L_3 \sin(\alpha_p + k_p + h_p) + L_4 \sin(\alpha_p + k_p + h_p + t_p) = 0 \end{aligned} \quad (7)$$

After dividing equation (7) by $\sin(\alpha_p)$, we get

$$\begin{aligned} &\Rightarrow L_1 + L_2 \cos(k_p) + L_2 \cot(\alpha_p) \sin(k_p) + L_3 \cos(k_p + h_p) + L_3 \cot(\alpha_p) \sin(k_p + h_p) \\ &\quad + L_4 \cos(k_p + h_p + t_p) + L_4 \cot(\alpha_p) \sin(k_p + h_p + t_p) = 0 \\ &\Rightarrow \cot(\alpha_p) = -\frac{L_1 + L_2 \cos(k_p) + L_3 \cos(k_p + h_p) + L_4 \cos(k_p + h_p + t_p)}{L_2 \sin(k_p) + L_3 \sin(k_p + h_p) + L_4 \sin(k_p + h_p + t_p)} \\ &\Rightarrow \alpha_p = -\tan^{-1}\left(\frac{L_2 \sin(k_p) + L_3 \sin(k_p + h_p) + L_4 \sin(k_p + h_p + t_p)}{L_1 + L_2 \cos(k_p) + L_3 \cos(k_p + h_p) + L_4 \cos(k_p + h_p + t_p)}\right) \end{aligned} \quad (8)$$

From Table III, it can be seen that ankle pitch lies between the range of $(-67.97^\circ, 53.4^\circ)$. So, we constrain the calculated ankle pitch within these joint limits. After assigning the values of joint limits if the $\text{Real}(Z_0) = \text{Real}(\overrightarrow{ACoM})$ is within $[-3 \text{ cm}, 9 \text{ cm}]$ that is CoM is inside the foot base, then no further changes are required and the angles are published in the ROS.

$$\begin{aligned} \text{Real}(Z_0) &= L_1 \cos(90 + \alpha_p) + L_2 \cos(90 + \alpha_p + k_p) + L_3 \cos(90 + \alpha_p + k_p + h_p) \\ &\quad + L_4 \cos(90 + \alpha_p + k_p + h_p + t_p) = 0 \end{aligned}$$

$$\text{Real}(Z_0) \in [-3, 9]$$

However, if $\text{Real}(Z_0)$ exceeds its limits, then we adjust the knee pitch to balance the robot.

Knee Pitch: There are various motions that a human can do with ease, but the robot can fail to perform with just mapping of joint angles and controlling ankle pitch. To perform such actions, we need to adjust the incoming knee pitch slightly to keep the robot balanced. From Fig. 2 and equation (7), knee pitch is

Table III. NAO leg joints limits for pitch control.

Right leg joint angles limits	Left leg joint angles limits
$h_p = (-88^\circ, 27.73^\circ)$	$h_p = (-88^\circ, 27.73^\circ)$
$k_p = (-5.9^\circ, 121.47^\circ)$	$k_p = (-5.29^\circ, 121.04^\circ)$
$\alpha_p = (-67.97^\circ, 53.4^\circ)$	$\alpha_p = (-68.15^\circ, 52.86^\circ)$

derived as given below

$$\begin{aligned} &\Rightarrow L_1 \sin(\alpha_p) + L_2 \sin(\alpha_p + k_p) + L_3 \sin(\alpha_p + k_p + h_p) + L_4 \sin(\alpha_p + k_p + h_p + t_p) = 0 \\ &\Rightarrow L_1 \sin(\alpha_p) + L_2 \sin(\alpha_p) \cos(k_p) + L_2 \cos(\alpha_p) \sin(k_p) + L_3 \sin(k_p) \cos(\alpha_p + h_p) \\ &\quad + L_3 \cos(k_p) \sin(\alpha_p + h_p) + L_4 \sin(k_p) \cos(\alpha_p + h_p + t_p) + L_4 \cos(k_p) \sin(\alpha_p + h_p + t_p) = 0 \\ &\Rightarrow [L_2 \sin(\alpha_p) + L_3 \sin(\alpha_p + h_p) + L_4 \sin(\alpha_p + h_p + t_p)] \cos(k_p) \\ &\quad + [L_2 \cos(\alpha_p) + L_3 \cos(\alpha_p + h_p) + L_4 \cos(\alpha_p + h_p + t_p)] \sin(k_p) = -L_1 \sin(\alpha_p) \end{aligned} \tag{9}$$

Let,

$$\begin{aligned} A &= [L_2 \cos(\alpha_p) + L_3 \cos(\alpha_p + h_p) + L_4 \cos(\alpha_p + h_p + t_p)], \\ B &= [L_2 \sin(\alpha_p) + L_3 \sin(\alpha_p + h_p) + L_4 \sin(\alpha_p + h_p + t_p)], \text{ and } C = -L_1 \sin(\alpha_p) \\ &\Rightarrow A \sin(k_p) + B \cos(k_p) = C \end{aligned}$$

$$\Rightarrow \frac{A}{\sqrt{A^2 + B^2}} \sin(k_p) + \frac{B}{\sqrt{A^2 + B^2}} \cos(k_p) = \frac{C}{\sqrt{A^2 + B^2}}$$

let, $\frac{A}{\sqrt{A^2 + B^2}} = \sin(\phi)$, and $\frac{B}{\sqrt{A^2 + B^2}} = \cos(\phi)$

$$\Rightarrow \sin(k_p) \sin(\phi) + \cos(k_p) \cos(\phi) = \frac{C}{\sqrt{A^2 + B^2}} \tag{10}$$

$$\Rightarrow \cos(k_p - \phi) = \frac{C}{\sqrt{A^2 + B^2}}$$

$$\Rightarrow k_p = \pm \cos^{-1}\left(\frac{C}{\sqrt{A^2 + B^2}}\right) + \phi + 2\pi n, \text{ where } n \in \mathbb{Z} \ \& \ \phi = \tan^{-1}\left(\frac{A}{B}\right)$$

$$\Rightarrow k_p = \pm \cos^{-1}\left(\frac{C}{\sqrt{A^2 + B^2}}\right) + \tan^{-1}\left(\frac{A}{B}\right) + 2\pi n, \text{ where } n \in \mathbb{Z} \tag{11}$$

Hip pitch: The above formula is used to find the knee pitch when ankle pitch alone cannot suffice the balancing criteria. However, there may be a situation when both ankle pitch and knee pitch together cannot balance the robot. For that, we need to slightly adjust the hip pitch to balance the robot. Then from Fig. 2,

$$\Rightarrow \sin(\alpha_p + k_p + h_p) = -\frac{L_1 \sin(\alpha_p) + L_2 \sin(\alpha_p + k_p) + L_4 \sin(t_p)}{L_3} \tag{12}$$

$$\Rightarrow h_p = -\sin^{-1}\left(\frac{L_1 \sin(\alpha_p) + L_2 \sin(\alpha_p + k_p) + L_4 \sin(t_p)}{L_3}\right) - \alpha_p - k_p \tag{13}$$

The overall procedure of pitch control to balance the whole body is presented in Algorithm 1. It aims to compute α_p, k_p, h_p such that $Real(Z_0)$ always lies in range $[-3, 9]$ cm. The $constrain(j_angle, Min, Max)$ function constrains the given joint angles j_angle within nao joint limits. Min is the lower limit, and Max is the upper limit of a particular joint. For both legs, the hip pitch (h_p) and knee pitch (k_p) from the motion file are used to calculate ankle pitch (α_p) from

Algorithm 1. Proposed Pitch control algorithm

```

1: Procedure constrainj_angle, Min, Max
2:   if j_angle < Min then
3:     j_angle ← Min
4:   else
5:     if j_angle > Max then
6:       j_angle ← Max
7:   return j_angle
8: Procedure calcAnklePitchhp, kp, tp
9:   return arctan ((L2 sin (kp) + L3 sin (kp + hp) + L4 sin (kp + hp + tp))/(- 1 * (L1 + L2 cos (kp)
+L3 cos (kp + hp) + L4 cos (kp + hp + tp)))
10: Procedure calcKneePitchαp, hp
11:   A = L2 cos (αp) + L3 cos (αp + hp) + L4 cos (αp + hp + tp)
12:   B = L2 sin (αp) + L3 sin (αp + hp) + L4 sin (αp + hp + tp)
13:   C = -L1 sin (αp)
14:   return arccos (C/√A2 + B2) + arctan (A/B)
15: Procedure calcHipPitchαp, kp, tp
16:   return arcsin (- 1 * (L1 sin (αp) + L2 sin (αp + kp) + L4 sin (tp))/L3) - αp - kp
17: for i = 0 to 2 do
18:   αp = calcAnklePitch(hp, kp, tp)
19:   αp = constrain(αp, αMin, αMax)
20:   if (RealZMin < RealZ(αp, hp, kp, tp) < RealZMax) then
21:     publish αp, hp, kp
22:   else
23:     kp = calcKneePitch(αp, hp, tp)
24:     kp = constrain(kp, kMin, kMax)
25:     if (RealZMin < RealZ(αp, hp, kp, tp) < RealZMax) then
26:       publish αp, hp, kp
27:     else
28:       hp = calcHipPitch(αp, kp, tp)
29:       hp = constrain(hp, hMin, hMax)
30:       if (RealZMin < RealZ(αp, hp, kp, tp) < RealZMax)
31:         publish αp, hp, kp
32:
33: exit

```

the function $calcAnklePitch(h_p, k_p, t_p)$. The computed ankle pitch is checked whether it is in its joint limits through the $constrain(j_angle, a_{Min}, a_{Max})$ function. If the ankle pitch is less than a_{Min} (lower limit of ankle pitch) given in Table III; then, ankle pitch is assigned to $\alpha_p = a_{Min} = -67.97$ (for right leg as an example), and if ankle pitch is greater than a_{Max} (upper limit of ankle pitch), then it is assigned to $\alpha_p = a_{Max} = 53.4$. Then if the $Real(Z_0) = RealZ(\alpha_p, h_p, k_p, t_p)$ is in range of $[-3, 9]$ cm where $RealZ_{Min} = -3$ and $RealZ_{Max} = 9$, then the angles are published in ROS, otherwise knee pitch is calculated from $calcKneePitch(\alpha_p, h_p, t_p)$ function. The calculated knee pitch is constrained within joint limits as given in Table III by the function $constrain(k, k_{Min}, k_{Max})$. And checked if the $Real(Z_0)$ is in range of $[-3, 9]$ cm; then, the angles are published in ROS. Otherwise, hip pitch is calculated from the function $calcHipPitch(\alpha_p, k_p, t_p)$; it is also constrained within its joint limits and $Real(Z_0)$ is again checked for $[-3, 9]$ cm range. If it is in the range then joint angles are published otherwise exit.

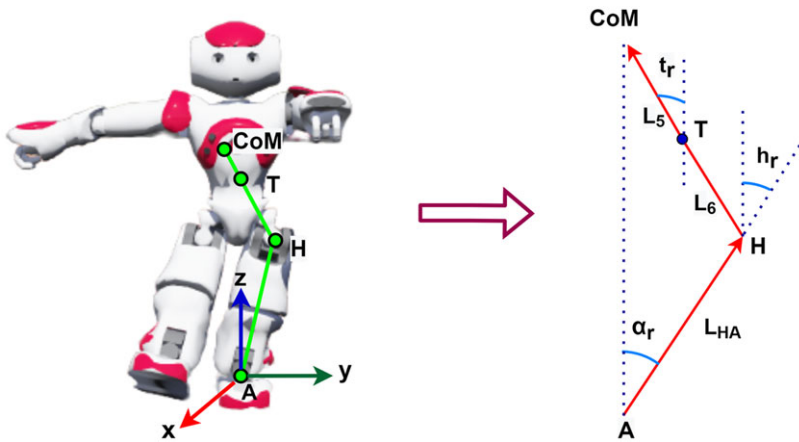


Figure 3. A front view representation of left leg joints of NAO robot in one-leg stand position.

3.2.2 Roll control

Similar to pitch control, the robot also needs roll control to balance itself in a one-leg stand posture. The algorithm predicts the position of CoM and also ensures that it remains inside the single foot base area. It controls the ankle roll and hip roll to maintain the whole-body balance of the NAO robot.

Ankle roll: The motions involving single leg stand is difficult and challenging to balance as compared to motions with double leg stand. The ankle roll is a function of L_{HA} that is the length of Hip to ankle and it depends on the ankle pitch and knee pitch. This is because of the distance between hip and ankle changes when the knee bends. The Fig. 3 represents the front view of the robot where it is balancing on its left foot. All the lengths between the joints are taken from TF (transforms). So, from Fig. 3 the ankle roll (α_r) is written as

$$\overrightarrow{ACoM} = \overrightarrow{AH} + \overrightarrow{HT} + \overrightarrow{TCoM} \tag{14}$$

$$Real(\overrightarrow{ACoM}) = 0$$

$$L_{HA} \sin(\alpha_r) + L_6 \sin(h_r) + L_5 \sin(t_r) = 0 \tag{15}$$

where, $t_r = \tan^{-1} \left(\frac{y}{z} \right)$ & $L_5 = \sqrt{y^2 + z^2}$

$$\alpha_r = \sin^{-1} \left(\frac{L_6 \sin(h_r) + L_5 \sin(t_r)}{L_{HA}} \right) \tag{16}$$

A side view representation of right leg joints of NAO robot in one-leg stand position is given in Fig. 4. The imaginary part of the Z_1 vector is considered to get the vertical projection of the leg. The ankle pitch (α_p) and knee pitch (k_p) are calculated from the above pitch control algorithm. Then from Fig. 4,

$$\begin{aligned} Z_1 &= \overrightarrow{AK} + \overrightarrow{KH} \\ &= L_1 \angle(90^\circ + \alpha_p) + L_2 \angle(90^\circ + \alpha_p + k_p) \\ L_{HA} &= Img(Z_1) \\ &= L_1 \sin(90^\circ + \alpha_p) + L_2 \sin(90^\circ + \alpha_p + k_p) \\ &= L_1 \cos(\alpha_p) + L_2 \cos(\alpha_p + k_p) \end{aligned}$$

Table IV. NAO leg joints limits for roll control.

Right leg joint angles limits	Left leg joint angles limits
$\alpha_r = (-44.06^\circ, 22.80^\circ)$	$\alpha_r = (-22.79^\circ, 44.06^\circ)$
$h_r = (-45.29^\circ, 21.74^\circ)$	$h_r = (-21.74^\circ, 45.29^\circ)$

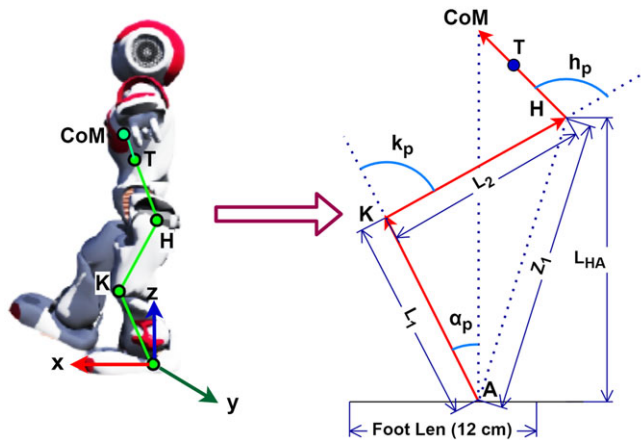


Figure 4. A side view representation of left leg joints of NAO robot in one-leg stand position.

Putting the value of L_{HA} in equation (16),

$$\alpha_r = -\sin^{-1}\left(\frac{L_6 \sin(h_r) + L_5 \sin(t_r)}{L_1 \cos(\alpha_p) + L_2 \cos(\alpha_p + k_p)}\right) \tag{17}$$

If $\text{Real}(\rightarrow \text{ACoM}) \neq 0$, then hip roll is calculated by the formula given in equation (18),

$$\begin{aligned} h_r &= -\sin^{-1}\left(\frac{L_{HA} \sin(\alpha_r) + L_5 \sin(t_r)}{L_6}\right) \\ &= -\sin^{-1}\left(\frac{(L_1 \cos(\alpha_p) + L_2 \cos(\alpha_p + k_p)) \sin(\alpha_r) + L_5 \sin(t_r)}{L_6}\right) \end{aligned} \tag{18}$$

The plot of joint angles of a motion when the left leg is on the ground and the right leg is moving sideways is shown in Fig. 5. From the Table IV, we can see that the right and left hip roll (h_r) have dissimilar ranges. So, it is inverted by multiplying with -1 to map them in the same range and ease the calculations. Thereafter, the difference of the right and left hip roll is taken to predict which leg should balance the robot and which leg should perform the action. While performing the motion, if the difference exceeds a threshold angle 2° (determined experimentally), the left hip roll is overridden to balance the robot. In Fig. 6, the red dotted line represents the threshold region. If the difference of LHipRoll and RHipRoll (red line) is greater than the positive threshold value, the right leg will balance and the left leg will perform the given motion. When the difference is lower than the negative threshold value, the left leg will balance and the right leg will perform the motion. The LHipRoll line in Fig. 6 represents that the joint angle values are overridden by the proposed algorithm to balance the robot while performing a motion in which left leg is on the ground and right leg is moving sideways.

The Algorithm 2 explains the proposed roll control mechanism for balancing the whole body of NAO robot. The *constrain(j_angle, Min, Max)* function constrains the given joint angles *j_angle* within NAO joint limits. *Min* is the lower limit, and *Max* is the upper limit of a particular joint. Here, *LHR* & *RHR* are the left and right hip roll taken from the motion file to calculate *balanceLeg* that is on which leg the robot should balance. If the *balanceLeg* is lower than negative *threshValue* which is the threshold

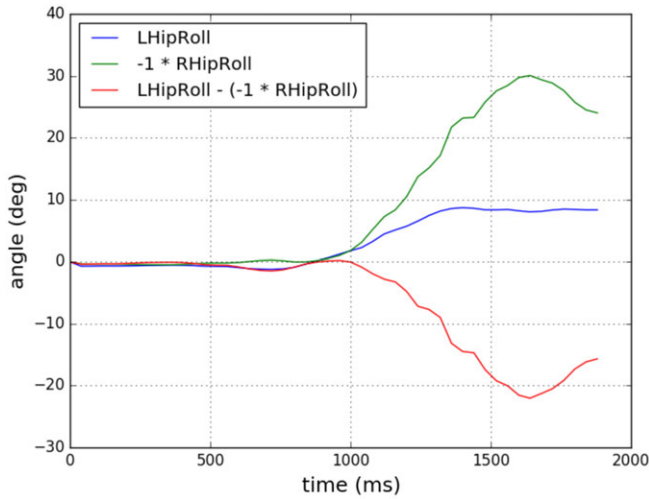


Figure 5. A plot of raw joint angles when the NAO’s left leg is on the ground and right leg is moving to the side.

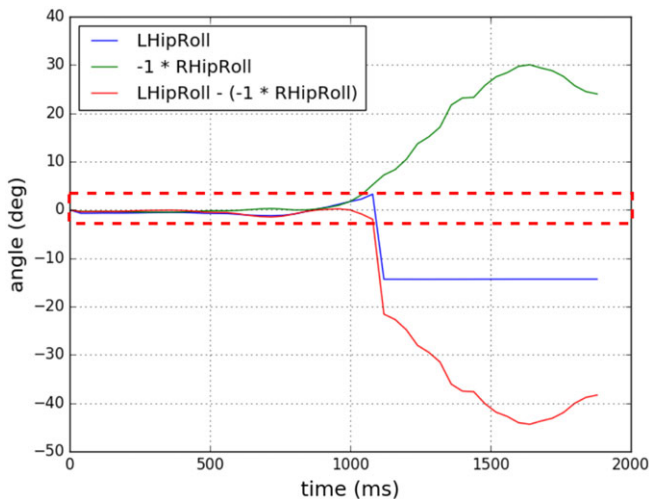


Figure 6. A plot of processed joint angles after implementing the proposed roll control algorithm to balance the whole body of NAO when the left leg is on the ground and right leg is moving to the side.

determined experimentally as 2° , then the *balanceLeg* will be the left leg and the right leg will perform the given motion. When the *balanceLeg* is greater than positive *threshValue*, *balanceLeg* will be the right leg and the left leg will perform the given motion. Ankle roll (α_r) is calculated by taking the ankle pitch (α_p), knee pitch (k_p), and hip roll (h_r) from the motion file. The function *constrain*($\alpha_r, a_{Min}, a_{Max}$) is used to constrain the ankle roll within its lower limit a_{Min} & upper limit a_{Max} given in Table IV according to the left or right *balanceLeg*. Then if the $Real(Z_0) = RealZ(\alpha_r, h_r, t_r)$ is in range of $[-3, 9]$ cm where $RealZ_{Min} = -3$ and $RealZ_{Max} = 9$, then the angles are published in ROS; otherwise, hip roll is calculated. The *constrain*(h_r, h_{Min}, h_{Max}) is used for assigning the hip roll between the lower joint limit h_{Min} and upper joint limit h_{Max} of hip given in Table IV. And checked if the $Real(Z_0)$ is in range of $[-3, 9]$ cm, then the hip roll and ankle roll are published to the ROS for whole-body balance otherwise exit.

Algorithm 2. Proposed Roll control algorithm

```

1: Procedure constrainj_angle, Min, Max
2:   if j_angle < Min then
3:     j_angle ← Min
4:   else
5:     if j_angle > Max then
6:       j_angle ← Max
7:   return j_angle
8: balanceLeg ← LHR - (-1 * RHR)
9: if balanceLeg < -threshValue then
10:  balanceLeg ← leftLeg
11: else
12:  if balanceLeg > threshValue then
13:    balanceLeg ← rightLeg
14: Img(Z1) ← L1 * cos(ap) + L2 * cos(ap + kp)
15: αr ← arcsin((-1) * L6 sin(hr) + L5 sin(tr) / Img(Z1))
16: αr = constrain(αr, aMin, aMax)
17: if (RealZMin < RealZ(αr, hr, tr) < RealZMax)
18:  publish αr, hr
19: else
20:  hr ← arcsin((-1) * Img(Z1) * sin(αr) + L5 sin(tr) / L6)
21:  hr = constrain(hr, hMin, hMax)
22:  if (RealZMin < RealZ(αr, hr, tr) < RealZMax) then
23:    publish αr, hr
24:  else
25:    exit

```

3.3 Human motion recognition using CNN

The concept of robots recognising the human motions has become essential for reliable teleoperation. The prerequisite for this system is to have an ability to predict the intentions of human operators to perform a meaningful task. One of the major challenges of human motion recognition is the human motion variability. The same motion is performed differently by the same person over time and also different person perform the same motion in different ways. In this case, the robot should efficiently recognise the human motions and perform the tasks associated with it. To handle this issue, a novel deep 1D-CNN is presented in this work. The architecture of the proposed 1D-CNN consists of 7-layer deep network structure represented in Fig. 7. It comprises four convolutional layers, one max-pooling layer, one global average pooling layer, and one fully connected layer. The proposed 1D-CNN is designed for the automated classification of whole-body human motions.

Zero Padding: The foremost challenge comes while dealing with 1D CNN is the length of the input samples must be same. But in human motion recognition, the motions captured by Kinect may have different durations. Hence, the length of input motions are different. It is necessary to adopt a preprocessing technique which converts the input samples of different lengths into a fixed length. Therefore, zero padding is introduced to satisfy the condition imposed by the input layer of CNN. Zero can be padded at the start or end of the input vector. It can also be padded both at the start and end of the input vector by keeping the data values in the middle of the vector. Here, the input samples are mapped joint angles which are fed to the proposed 1D-CNN. The input sample with the maximum length (max_len) is determined. The length of the remaining input samples is increased to the max_len by padding zero so that each sample has an equal length.

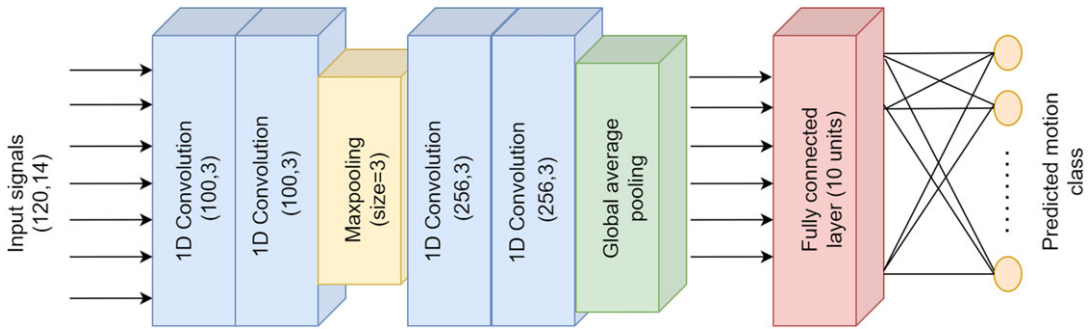


Figure 7. Block representation of the proposed deep 1D-CNN architecture.

Proposed deep 1D-CNN Architecture: The first layer is the convolution layer that extracts features from the given fixed-length input signals of shape 120×14 . The convolution operations is performed using 100 filters with a kernel size of 3 on each input vector with $\text{stride} = 1$. The convoluted joint angle signals of n frames are used to generate feature maps containing the most prominent features. In the second layer, the convolution operation is repeated with the same 100×3 size on the previously obtained feature maps with $\text{stride} = 1$. In the third layer, max-pooling layer is used to generate new feature maps by taking the maximum values in the specified regions on the feature maps obtained from the previous convolution layer. It reduces the size of feature maps according to the pooling size. In this layer, the feature maps are reduced by one third using 3-unit regions. In the fourth and fifth layer, the convolution operation is repeated with 256×3 size combinations to learn higher-level features. The Rectified Linear Unit (RELU) is used as an activation function in all the convolution layers to impart non-linearity in the network structure. The output of the fifth layer is fed to the global average pooling layer. It generates new feature maps by taking the average values in the specified regions of the feature maps obtained from the previous layers. It also helps to avoid overfitting at this layer by reducing the parameters in the model. The features obtained from the global average pooling layer are placed as the input to the fully connected layer. This layer connects each neuron of the layer to each neuron of the next layer. The dropout rate of 0.8 is applied in this layer. It loses the 20% of hidden layer neurons; thus, dependency between the neurons is reduced. It is equivalent to training on different NNs with the reduced neurons and helps to learn more robust and diverse features. This is applied to avoid overfitting and improve generalisation during the learning phase in the NN. The output of the fully connected layer is fed to the softmax function, which performs the classification of human motions. It computes the probability of each motion by mapping the output of neurons into $[0, 1]$ intervals. The motion with the highest probability is the predicted human motion. The detailed parameters of each layer of the deep 7-layer 1D-CNN are given in Table V. The conventional backpropagation algorithm with a batch size of 121 is used to train the CNN model for 1000 epochs. The Adaptive moment estimation (Adam) is the optimisation algorithm adopted to update the parameters of the proposed 1D-CNN with a learning rate of 0.0001. It enhances the efficiency of the training process and converges at a faster rate. The categorical cross-entropy loss function is used, given that the model learning is a multi-class classification problem. All the chosen parameters of proposed 1D-CNN are determined by brute force technique after observing the performances of the validation sets.

4. Experimental results

In this section, the experimental results of the proposed framework are presented. The developed deep 1D-CNN model automatically classifies ten different whole-body motions. The dataset is split into three parts: training, validation, and test sets. In this experimental study, 60% of the data is used for the training set, 20% is used as a validation set, and the remaining 20% is used as a test set. These distributions are

Table V. The detailed parameters of the proposed deep 1D-CNN model.

Layer no.	Layer type	Unit × Kernel	Layer parameters	Output shape	No. of parameters
1	Convolution	100 × 3	Stride = 1, Activation=RELU	118 × 100	4300
2	Convolution	100 × 3	Stride = 1, Activation=RELU	116 × 100	30,100
3	Max-pooling	3	Stride = 3	38 × 100	0
4	Convolution	256 × 3	Stride = 1, Activation=RELU	36 × 256	77,056
5	Convolution	256 × 3	Stride = 1, Activation=RELU	34 × 256	196,864
6	Global average pooling	–	–	256	0
7	Fully-connected	–	Activation= Softmax, Dropout rate = 0.8	10	2570

randomly selected. The training set is used to train the CNN model, and the validation set is used to tune the CNN parameters. The trained model is then applied on the testing set. The data in the testing phase are never seen by the model in the learning phase. The experiment performed is a supervised learning process. The training and validation performance graphs of the proposed 1D-CNN model during 1000 epochs are shown in Fig. 8(a) and (b). It is clearly seen from the graphs that there is no overfitting problem in the proposed model. The training accuracy attains 98.30% in the training phase and the validation accuracy achieves 95%. The training loss value in the learning phase is 0.0170, and the validation loss value of the model is 0.760. The critical performance achieved by the trained model after applying on the test data is analysed with standard evaluation criteria. The criteria used to assess the proposed model performance are precision, sensitivity, specificity, and $F1$ score. The overall accuracy of human motion recognition is also calculated for developed 1D CNN, NN, and HMM. The graphical representation for comparison of the evaluation metrics of the proposed CNN model and state-of-the-art methods is presented in Fig. 9. Ajili et al. [29] used HMM to develop a gesture recognition system for nao robot teleoperation. The joint angle features were extracted by applying Laban movement analysis. The extracted features were fed into the HMM. A grid search between 5 to 100 was done to find the optimum number of symbols and states. The best recognition results obtained with the state = 5 and observation symbols = 12. All the predicted gestures are converted into commands and sent to the NAO robot. The developed HMM-based gesture recognition system is implemented on our dataset and attains an accuracy of 89.7%. Jawed et al. [43] developed a NN based system to classify the whole-body postures of the patients for rehabilitation. The human skeleton was tracked by placing the Kinect at 0.6 m distance and 1.4 m height. The joint angles were extracted from the skeleton joint points and fed to the classifier. The NN-based recognition system is applied on our dataset and achieves an accuracy of 92.3%. The proposed 7-layer CNN model provides 95% recognition performance, which is more than the overall accuracy attained by the NN and HMM.

The judicious selection of model parameters plays a key role in achieving high recognition performances. It may cause an overfitting or underfitting problem with inappropriate parameters. To analyse the performance of 1D-CNN, different sizes of convolutional kernels are taken into consideration in our experimental study. The effect of changing the filter size on the proposed model performance is presented in Fig. 10(a). The model performance is tested for the kernel size ranging from 3 to 10. The highest performance is achieved at kernel size = 3. The graph illustrates that the selection of the right kernel size is very important as it affects the performance of the model drastically. The effect of changing pool size on the model performance is shown in the Fig. 10(b). It concludes that it does not contribute

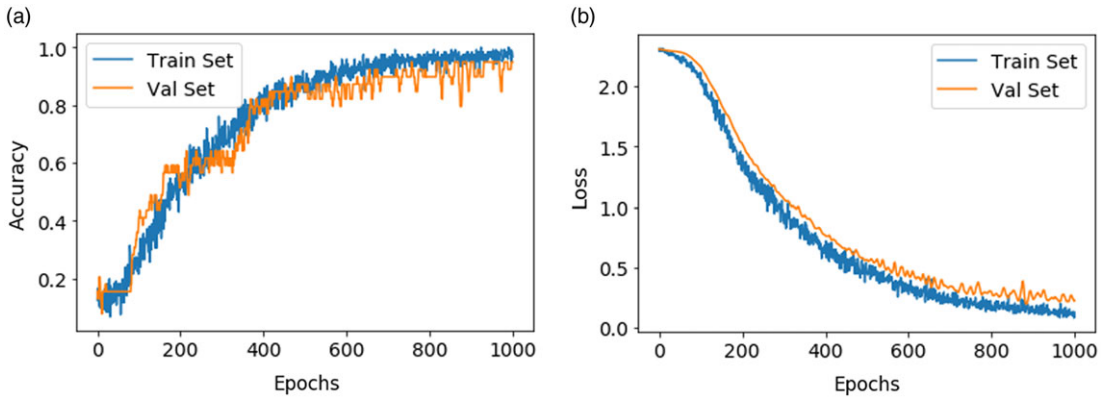


Figure 8. Training and validation performances of the proposed deep 1D-CNN model with number of epochs (a) Model accuracy (b) Model loss.

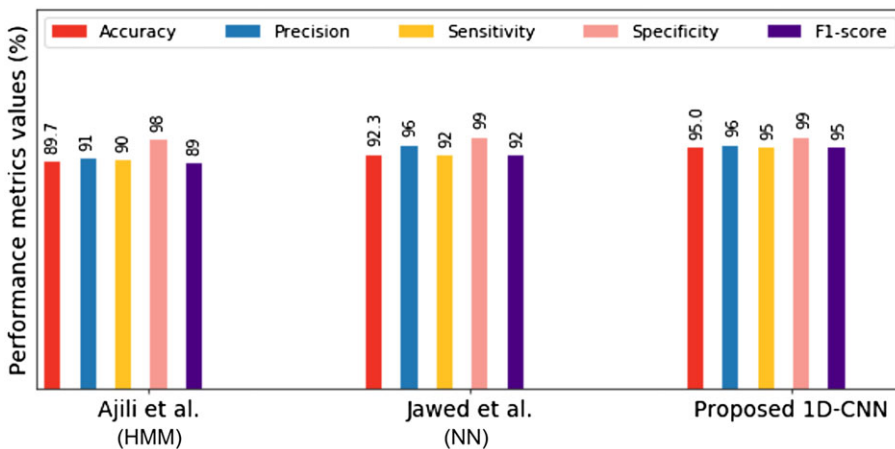


Figure 9. Detailed classification performance of the proposed model and other state-of-the-art methods.

much in increasing the model performances. It can also be seen from the graph that if we keep increasing the pool size, then the recognition performance reduces in our case. Based on the multiple runs, pooling size of 3 is found to be best for the recognition. The performance of the model obtained with different learning rate is presented in Fig. 10(c). The learning rate affects the recognition performances of the model more than the pooling size. In our case, small changes in the learning rate sometimes causing overfitting of the model. The best result is obtained with learning rate of 0.0001. The selection of optimisers has significant effects on the recognition performances of the classification system as shown in Table VI. The Stochastic Gradient Descent (SGD) with momentum and RMSProp optimiser has almost same recognition performance in our dataset. The Nadam optimiser provided a good recognition performance but not as good as AdaMax. The highest recognition is achieved by the Adam optimiser. The proposed 1D-CNN with selected model parameters is less complex and simpler to use. Moreover, it performs better recognition than the state-of-the-art methods. The motion predicted by the proposed model is published to ROS node. The NAO robot subscribes the defined node and imitates the same motion. For example, a human is performing a motion in which both hands are raised on the side and left leg is also on the side as shown in Fig. 11. The NAO robot is imitating the same human motion efficiently with a balance control.

Table VI. Performances of the proposed deep 1D-CNN classification system with different optimisers.

Optimisers	Accuracy of the proposed 1D-CNN model (%)
AdaMax	94.54
Nadam	92.72
RMSProp	89.09
Adam	95
SGD + momentum	89

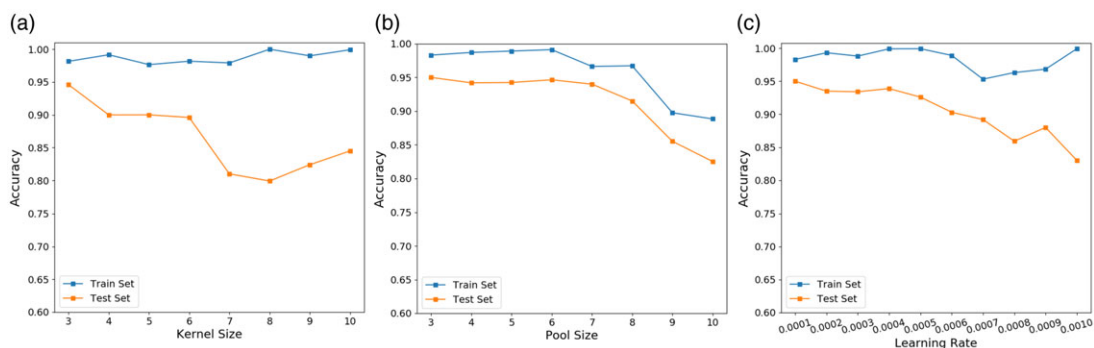


Figure 10. Training and testing performances of the proposed deep 1D-CNN model with (a) increasing kernel size (b) increasing pooling size and (c) increasing learning rate.

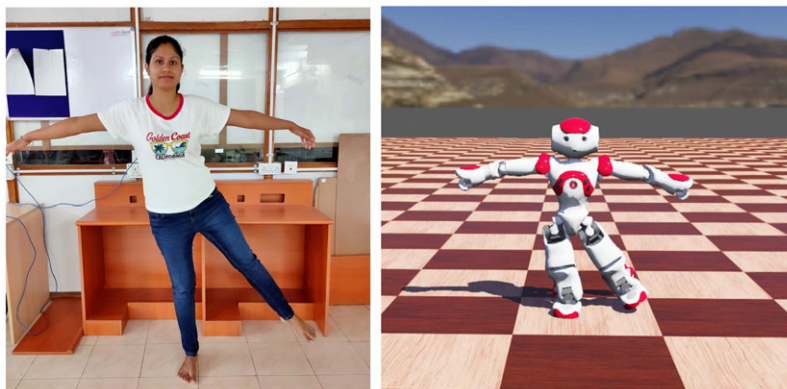


Figure 11. Imitation of human motion by NAO robot.

If the motion information provided by the Kinect is applied directly to the robot then the robot may not perform the task accurately due to noise, experimental error, robot joint constraints, tracking error etc. Also, if a human performs a slightly different motion, then the robot may confuse or recognise the wrong motion, thus performing the wrong task. Furthermore, it will cause a problem for balance control. The proposed teleoperation framework provides a good solution in this scenario. To tackle this issue, the proposed 1D-CNN is trained with motions of different persons. So that the robot learns the way of performing the same gesture by the different persons. A motion library is created with motion names and its corresponding best motion information. Using the proposed recognition approach, it recognises the motions accurately and picks its best motion information to imitate the motion. Once it predicts the motion, the robot automatically prepares itself to balance its body and imitates the motions precisely.

5. Conclusions and future work

In this study, a teleoperation framework is proposed based on Kinect and a humanoid robot, with the capability of motion recognition and balance control. The proposed adaptive balancing technique for NAO facilitates the robot to complete full-body postures efficiently. It works effectively in single as well as double supporting phases. It estimates the dynamic CoM by solving forward kinematics of the humanoid robot and applying weighted average of mass with the CoMs of individual links with respect to the previous joint frames. It targets to keep the dynamic CoM inside the stable region, that is, the foot base of the robot. We developed a robust 1D-CNN-based model that enables the robot to recognise the full-body motions of humans. It uses the joint angles mapped from 3D skeleton information and performed with a high accuracy. After recognising the motions performed in front of the Kinect, the NAO robot automatically implements the proposed balancing technique and imitates the same motions. The proposed framework is reasonable and presents a novel paradigm for future research studies. It brings up new opportunities to build frameworks that easily integrate sensor-based techniques that produce signals such as EEG, EMG. It encourages gesture/motion-based natural interactions between human and robot. Our future work aims to extend the balance strategy to stabilise the robot dynamically under the effect of external forces and moments using inertial measurement units and model-based closed-loop feedback control system algorithms. Further, we will design advance features to facilitate the robots to understand the human intentions.

Conflicts of Interest. The authors declare that they have no conflict of interest.

Financial Support. None.

Ethical Considerations. None.

References

- [1] O. Alquisiris-Quecha, A. E. Maldonado-Reyes, E. F. Morales and L. E. Sucar, "Teleoperation and Control of a Humanoid Robot Nao through Body Gestures," *2018 Seventeenth Mexican International Conference on Artificial Intelligence (MICAI)* (IEEE, 2018) pp. 88–94.
- [2] J. Mi, Y. Sun, Y. Wang, Z. Deng, L. Li, J. Zhang and G. Xie, "Gesture Recognition Based Teleoperation Framework of Robotic Fish," *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2016) pp. 137–142.
- [3] J. Yan, K. Huang, K. Lindgren, T. Bonaci and H. J. Chizeck, Continuous operator authentication for teleoperated systems using hidden markov models. arXiv preprint arXiv:2010.14006 (2020).
- [4] Y. Ishiguro, T. Makabe, Y. Nagamatsu, Y. Kojio, K. Kojima, F. Sugai, Y. Kakiuchi, K. Okada and M. Inaba, "Bilateral humanoid teleoperation system using whole-body exoskeleton cockpit tablis," *IEEE Rob. Autom. Lett.* **5**(4), 6419–6426 (2020).
- [5] J. Ramos and S. Kim, "Humanoid dynamic synchronization through whole-body bilateral feedback teleoperation," *IEEE Trans. Rob.* **34**(4), 953–965 (2018).
- [6] M. Hirschmanner, C. Tsiourti, T. Patten and M. Vincze, "Virtual Reality Teleoperation of a Humanoid Robot Using Markerless Human Upper Body Pose Imitation," *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)* (IEEE, 2019) pp. 259–265.
- [7] C. Vongchumyen, C. Bamrung, W. Kamnitra and A. Watcharapong, "Teleoperation of Humanoid Robot by Motion Capturing Using Kinect," *2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST)* (IEEE, 2018) pp. 1–4.
- [8] J. Chen, G. Wang, X. Hu and J. Shen, "Lower-body control of humanoid robot nao via kinect," *Multimedia Tools Appl.* **77**(9), 10883–10898 (2018).
- [9] X. Yin, Y. Lan, S. Wen, J. Zhang and S. Wu, "Natural uav tele-operation for agricultural application by using kinect sensor," *Int. J. Agric. Biol. Eng.* **11**(4), 173–178 (2018).
- [10] A. Sripada, H. Asokan, A. Warriar, A. Kapoor, H. Gaur, R. Patel and R. Sridhar, "Teleoperation of a Humanoid Robot with Motion Imitation and Legged Locomotion," *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)* (IEEE, 2018) pp. 375–379.
- [11] D. P. Losey, C. G. McDonald, E. Battaglia and M. K. O'Malley, "A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction," *Appl. Mech. Rev.* **70**(1), 010804 (2018).

- [12] I. B. Abdallah and Y. Bouteraa, "Gesture Control of 3DOF Robotic Arm Teleoperated by Kinect Sensor," *2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)* (IEEE, 2019) pp. 145–151.
- [13] J. Avalos and O. E. Ramos, "Real-Time Teleoperation with the Baxter Robot and the Kinect Sensor," *2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC)* (IEEE, 2017) pp. 1–4.
- [14] A. Balmik, M. Jha and A. Nandy, "NAO Robot Teleoperation with Human Motion Recognition," *Arab. J. Sci. Eng.* **47**(2), 1137–1146 (2022).
- [15] M. Syakir, E. S. Ningrum and I. A. Sulistijono, "Teleoperation Robot Arm Using Depth Sensor," *2019 International Electronics Symposium (IES)* (IEEE, 2019) pp. 394–399.
- [16] E. Yavsan and A. UÇar, "Gesture imitation and recognition using kinect sensor and extreme learning machines," *Measurement* **94**, 852–861 (2016).
- [17] M. Arduengo, A. Arduengo, A. Colomé, J. Lobo-Prat and C. Torras, A robot teleoperation framework for human motion transfer. arXiv preprint arXiv:1909.06278 (2019).
- [18] A. Jha and S. S. Chiddarwar, "Robot programming by demonstration using teleoperation through imitation," *Ind. Robot Int. J.* **44**(2), 142–154 (2017).
- [19] B. E. Khadir, J. Varley and V. Sindhvani, Teleoperator imitation with continuous-time safety. arXiv preprint arXiv:1905.09499 (2019).
- [20] E.-J. Rolley-Parnell, D. Kanoulas, A. Laurenzi, B. Delhaisse, L. Rozo, D. G. Caldwell and N. G. Tsagarakis, "Bi-Manual Articulated Robot Teleoperation Using an External RGB-D Range Sensor," *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (IEEE, 2018) pp. 298–304.
- [21] D. Maraj, A. Maraj and A. Hajzeraj, "Application Interface for Gesture Recognition with Kinect Sensor," *2016 IEEE International Conference on Knowledge Engineering and Applications (ICKEA)* (IEEE, 2016) pp. 98–102.
- [22] X. Li, "Human–robot interaction based on gesture and movement recognition," *Signal Process. Image Commun.* **81**, 115686 (2020).
- [23] M. Ababneh, H. Sha'ban, D. AlShalabe, D. Khader, H. Mahameed and M. AlQudimat, "Gesture Controlled Mobile Robotic Arm for Elderly and Wheelchair People Assistance Using Kinect Sensor," *2018 15th International Multi-Conference on Systems, Signals & Devices (SSD)* (IEEE, 2018) pp. 636–641.
- [24] J. Dong, Z. Xia, W. Yan and Q. Zhao, "Dynamic gesture recognition by directional pulse coupled neural networks for human-robot interaction in real time," *J. Visual Commun. Image Represent.* **63**, 102583 (2019).
- [25] M.-Y. Shieh, C.-Y. Wang, W.-L. Wu and J.-M. Liang, "Gesture recognition based human–robot interactive control for robot soccer," *Microsyst. Technol.* **27**, 1175–1186 (2018).
- [26] M.-Y. Cho and Y.-S. Jeong, "Human Gesture Recognition Performance Evaluation for Service Robots," *2017 19th International Conference on Advanced Communication Technology (ICACT)* (IEEE, 2017) pp. 847–851.
- [27] G. Zhu, L. Zhang, P. Shen and J. Song, "An online continuous human action recognition algorithm based on the kinect sensor," *Sensors* **16**(2), 161 (2016).
- [28] Z. Ahmad, K. Illanko, N. Khan and D. Androustos, "Human Action Recognition Using Convolutional Neural Network and Depth Sensor Data," *Proceedings of the 2019 International Conference on Information Technology and Computer Communications* (2019) pp. 1–5.
- [29] I. Ajili, M. Malleem and J.-Y. Didier, "Gesture Recognition for Humanoid Robot Teleoperation," *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)* (IEEE, 2017) pp. 1115–1120.
- [30] J. Liu, A. Shahroudy, G. Wang, L.-Y. Duan and A. C. Kot, "Skeleton-based online action prediction using scale selection network," *IEEE Trans. Pattern Anal. Mach. Intell.* **42**(6), 1453–1467 (2019).
- [31] L. Zhang, Z. Cheng, Y. Gan, G. Zhu, P. Shen and J. Song, "Fast Human Whole Body Motion Imitation Algorithm for Humanoid Robots," *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2016) pp. 1430–1435.
- [32] D. Zhi, T. E. A. de Oliveira, V. P. da Fonseca and E. M. Petriu, "Teaching a Robot Sign Language Using Vision-based Hand Gesture Recognition," *2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)* (IEEE, 2018) pp. 1–6.
- [33] G. Cicirelli, C. Attolico, C. Guaragnella and T. D'Orazio, "A kinect-based gesture recognition approach for a natural human robot interface," *Int. J. Adv. Rob. Syst.* **12**(3), 22 (2015).
- [34] N. Hu and L. Zheng, "Human Action Imitation System Based on Nao Robot," *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)* (IEEE, 2019) pp. 2261–2264.
- [35] J. Jang, D. Kim, C. Park, M. Jang, J. Lee and J. Kim, Etri-activity3D: A large-scale RGB-D dataset for robots to recognize daily activities of the elderly. arXiv preprint arXiv:2003.01920 (2020).
- [36] C. Li, Q. Zhong, D. Xie and S. Pu, "Skeleton-based Action Recognition with Convolutional Neural Networks," *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (IEEE, 2017) pp. 597–600.
- [37] L. Hu and J. Xu, "A Spatio-Temporal Convolutional Neural Network for Skeletal Action Recognition," *International Conference on Neural Information Processing* (Springer, 2017) pp. 377–385.
- [38] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj and D. J. Inman, 1D convolutional neural networks and applications: A survey. arXiv preprint arXiv:1905.03554 (2019).

- [39] Z. Zhang, Y. Niu, Z. Yan and S. Lin, “Real-time whole-body imitation by humanoid robots and task-oriented teleoperation using an analytical mapping method and quantitative evaluation,” *Appl. Sci.* **8**(10), 2005 (2018).
- [40] https://en.wikipedia.org/wiki/Affine_transformation.
- [41] <https://developer.softbankrobotics.com/nao-naoqi-2-1/nao-documentation/nao-technical-guide/nao-h25/h25-links>.
- [42] <https://developer.softbankrobotics.com/nao-naoqi-2-1/nao-documentation/nao-technical-guide/nao-h25/h25-masses>.
- [43] U. Jawed, A. Mazhar, F. Altaf, A. Rehman, S. Shams and A. Asghar, “Rehabilitation Posture Correction Using Neural Network,” *2019 4th International Conference on Emerging Trends in Engineering, Sciences and Technology (ICEEST)* (IEEE, 2019) pp. 1–5.