Mechatronics 55 (2018) 38–53

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Novel lightweight odometric learning method for humanoid robot localization

Saeed Saeedvand [a], Hadi S. Aghdasi [b,*], Jacky Baltes [c]

[a] Department of Information Technology, Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran
[b] Department of Computer Engineering, Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran
[c] Department of Electrical Engineering, National Taiwan Normal University, Taipei, Taiwan

ARTICLE INFO

Keywords:
Odometry
Humanoid robots' localization
Closed-loop walk engine
Inverse kinematic

ABSTRACT

Odometry as one of the inevitable parts of robot behavior control plays an essential role in the localization of humanoid robots. Calculating odometry on the humanoid robots is usually based on one or combination of the vision, laser scanner, magnetic sensor, and pressure sensors. Vision or laser scanner-based approaches require high computational power for analyzing visual information. Hence, these kinds of approaches are not suitable for all kind of the small humanoid robots. On the other hand, it is known that magnetic sensors have instability problems in different environments. Furthermore, calculating accurate dead reckoning (pure odometry) is very difficult because of the complex mechanical system of humanoid robots, the presence of many sources of uncertainty, and inaccuracy in motion executions such as foot slippage. Therefore, this paper presents a robust learning method to localize a humanoid robot named Lightweight Humanoid robot Odometric Learning method (LHOL). This method does not employ any vision, magnetic, additional pressure sensors, and laser scanners, and, therefore, eliminates dependencies to these sensors for the first time. The method's core learning is based on the artificial neural network (ANN) which uses kinematic computations, IMU (roll and pitch data), and the robot's actuators' internal present load data as input data. The proposed LHOL method proves high accuracy on a novel fully 3D printed kid-sized humanoid robot platform (ARC) with both open-loop and closed-loop walk engines at differently covered floors.

1. Introduction

One of the fundamental principles of odometric localization is using some form of velocity measurement to keep track of the robot with respect to its starting location. Odometric localization methods can be classified into two main categories: (I) Dead reckoning and (II) Visual Odometry (VO). (I) Dead reckoning, inertial navigation system or pure odometry, is a relative positioning technique that provides the position and orientation of the robot with respect to a given starting point [1,2]. In contrast to wheeled robots, dead reckoning calculation in humanoid robots is difficult. It is because of the presence of many sources of uncertainty and inaccuracy in motion executions such as foot slippage and uneven floor. However, the main advantage of the inertial navigation system is that it is self-contained; it does not require external references such as landmarks [3]. (II) VO estimates the location of the robot by tracking the apparent motion of visual features or calculating the distance and orientation of the robot from some predefined objects or landmarks. VO requires high computational power to analyze visual information, which is not suitable for the weak embedded systems of small robots. Also, VO is sensitive to environment lighting and usually requires controlled environment conditions. In [4] authors have compared the state-of-the-art visual odometry approaches comprehensively and concluded that most of the VO systems proposed in the existing literature fail or cannot efficiently work in outdoor environments with shadows and directional sunlight.

The magnetic sensor can be used to easily calculate the orientation of the robot in the environment, which is useful for robot localization. However, there are some problems associated with the magnetic sensors such as electromagnetic interference (EMI), calibration challenges in different environments, etc. Thus, these sensors have instability problems in environments with high EMI. Moreover, it is known that mankind does not have any magnetic sensor. Due to these issues, using magnetic sensors at RoboCup humanoid robot soccer league competitions has been prohibited since 2016 [5]. To deal with the above-mentioned VO problems, a combination of VO and dead reckoning has recently been proposed [6]. However, achieving an accurate dead reckoning calculation is an inevitable necessity. Because of the complex mechanical system of humanoid robots and presence of many sources of uncertainty

* Corresponding author.
E-mail addresses: saeedvand@tabrizu.ac.ir (S. Saeedvand), aghdasi@tabrizu.ac.ir (H.S. Aghdasi), jacky.baltes@ntnu.edu.tw (J. Baltes).

https://doi.org/10.1016/j.mechatronics.2018.08.007
Received 27 January 2018; Received in revised form 26 July 2018; Accepted 25 August 2018
Available online 26 October 2018
0957-4158/© 2018 Elsevier Ltd. All rights reserved.

and inaccuracy in motion execution such as foot slippage, dead reckoning is very difficult to calculate. Therefore, utilizing learning-oriented approaches along with data from IMU sensors and joints' present load in humanoid robots can be useful for achieving high accuracy at improving robot dead reckoning estimations.

Artificial neural network (ANN) is one of the most commonly used learning-oriented approaches for estimating software development effort [7]. Abbas Heiat has compared neural network estimation method to regression approach [8]. This study examined the prediction performance of multilayer perceptron and radial basis function neural networks to that of regression analysis. His work proved that the computed mean absolute percentage error (MAPE) of the neural network model was significantly smaller than that of the regression model.

In this paper, we propose a robust learning method to localize a humanoid robot named Lightweight Humanoid robot Odometric Learning (LHOL). The main contributions of this paper are: (i) Elimination of using additional pressure sensors, (ii) Elimination of using compass, (iii) Elimination of using additional laser scanner on the humanoid robots to calculate accurate odometry, and (iv) no increase in computational power requirements; in contrast to Visual Odometry (VO), which requires high computational power (we do not utilize VO). Therefore, our new approach provides high accuracy in comparison with other humanoid robots' localization approaches just by using traditional existing data which is available in most of the humanoid robots (lightweight). In other words, this is the first time an approach has been proposed for humanoid robots which with minimum requirements achieve high accuracy in odometry. Hence, this approach can be utilized in a wide range of applications for humanoid robots such as navigation, behavior control, task planning, combining with state-of-the-art Visual Odometry (VO), etc.

The LHOL method is based on the artificial neural networks and multi-layered perceptron (MLP) with a back propagation training algorithm. The learning network inputs of LHOL method are based on kinematic computations, an inertial measurement unit (IMU), roll and pitch data, and robot actuators' internal present load data. We implemented and tested LHOL method on a novel fully 3D printed ARC robot (first fully 3D printed kid-sized humanoid robot platform) which was developed at our Humanoid Robots and Cognitive Technology Research Lab at the University of Tabriz. The provided learning method proved high accuracy (total average error in all experiments less than 9 cm; an improvement of 81.22% in localization accuracy) on the ARC robot with both open-loop and closed-loop walk engines at different floors i.e. carpet and artificial grass.

The rest of the paper is organized as follows. Section 2 discusses literature review of localization of humanoid robots. Section 3 presents the detail of manufactured ARC robot. Section 4 discusses LHOL method which contains an introduction to ARC robot inverse kinematics and our implemented analytical solution to inverse kinematics, the implemented Closed-loop walk engine, odometry learning and the proposed learning method based on ANN. Section 5 provides the experimental result of the proposed method on a novel 3D printed ARC robot. Finally, Section 6 concludes the paper.

2. Literature review

Unlike wheeled mobile robots, dead reckoning estimation (pure odometry) has been extensively studied. However, dead reckoning on humanoid robots is completely different from wheeled mobile robots. This difference is because of the complex mechanical system and complex control algorithmic nature of bipedal humanoid robots, on the one hand, and the presence of many sources of uncertainty and inaccuracy in motion execution such as walking on the grass, uneven floors, foot slippage, etc.

The state-of-the-art visual odometry types, approaches, challenges, and their applications have been discussed at [4] comprehensively. Authors argue that VO is an inexpensive and alternative odometry technique which is more accurate than GPS, INS, wheel odometry, and sonar localization systems. They identified computational cost, light and imaging conditions (i.e., directional sunlight, shadows, image blur, and image scale/rotation variance) as the primary challenges in VO systems. Furthermore, they concluded that most of the VO systems proposed in the existing literature fail or cannot efficiently work in outdoor environments with shadows and direct sunlight. A new adaptive estimation algorithm to estimate the wheel robot position by fusing the monocular vision and odometry/AHRS sensors and utilizing the properties of a perspective projection is provided at [9]. This algorithm was implemented in parallel using graphics processing unit (GPU) to achieve real-time performance. However, not all kinds of GPU do support CUDA cores: it is only limited to GeForce GPUs series. In [10] authors have used SLAM and an EKF on a Nao robot with dead reckoning which makes use of Nao's pressure Force-Sensing Resistor (FSR) sensors to improve the support foot choice. They also used the actual measurements coming from the camera (head position and orientation reconstructed by a V-SLAM algorithm). In another work [11] authors validated a method for odometric localization on a humanoid NAO robot using a monocular camera, an IMU, joint encoders and foot pressure sensors. They used the prediction-correction paradigm of the Extended Kalman Filter, which integrated these sources of data. The correction step of the filter was used as measurements of the torso orientation, sensed by the IMU, and the head position, reconstructed by a VSLAM algorithm. Their work still has the VO drawbacks, and they do not take advantage of any learning algorithm. In [12] authors presented a strategy for achieving localization of a humanoid robot (using an NAO humanoid robot and a Kinect) in environments monitored by external devices. They used a particle filter method over depth images captured by an expensive RGB-D sensor to efficiently track the position and orientation of the robot during its march. They also presented a basic communication framework between the tracking and the locomotion control of the robot based on the robot's operating system with real-time locomotion tasks capabilities. In [6] authors computed an improved odometry estimate by fusing visual odometry, feedforward commands from gait generator and orientation from inertial sensors. They generated a 3D point cloud from the accumulation of successive laser scans, and the point cloud was, then, sliced to create a constant height horizontal virtual scan. So they used this slice as an observation base and fed to a 2D SLAM method. They tested the results on an adult-size JAXON robot. Based on this literature, considering laser scanners' high weight and large sizes, it can be concluded that laser scanner-based approaches require high computational power for analyzing visual information.

On the humanoid robots, quite a few works have been proposed for the use of dead reckoning to calculate or even improve total odometry accuracy. In [13] authors developed a step prediction model that estimated the location of the next footstep in Cartesian coordinates given that the same inputs control the central pattern generator. They used motion capture data recorded from walking robots to estimate the parameters of the prediction model and to verify the accuracy of the predicted footstep locations. They achieved a precision with a mean error of 103 cm which is very large and unacceptable error. In [14] authors proposed a method to compensate for odometry drifting using machine learning on a small size low-cost humanoid without vision. At this work, authors used additional foot pressure sensors made of four strain gauges and measured weight put on each cleat. Using these sensors requires a high cost. Moreover, their maintenance is difficult and increase system complexity. They predicted robot odometry through the Locally Weighted Projection Regression (LWPR) algorithm. Their work aimed at improving the standard internal odometry component. The results of their work on artificial grass using an open-loop walk engine showed an improvement from 24.5 cm to 10.3 cm (the average online sensor-based odometry drift), and from 62.0 cm to 14.8 cm after 10 seconds of walking (average off-line simulation-based odometry drift). They used

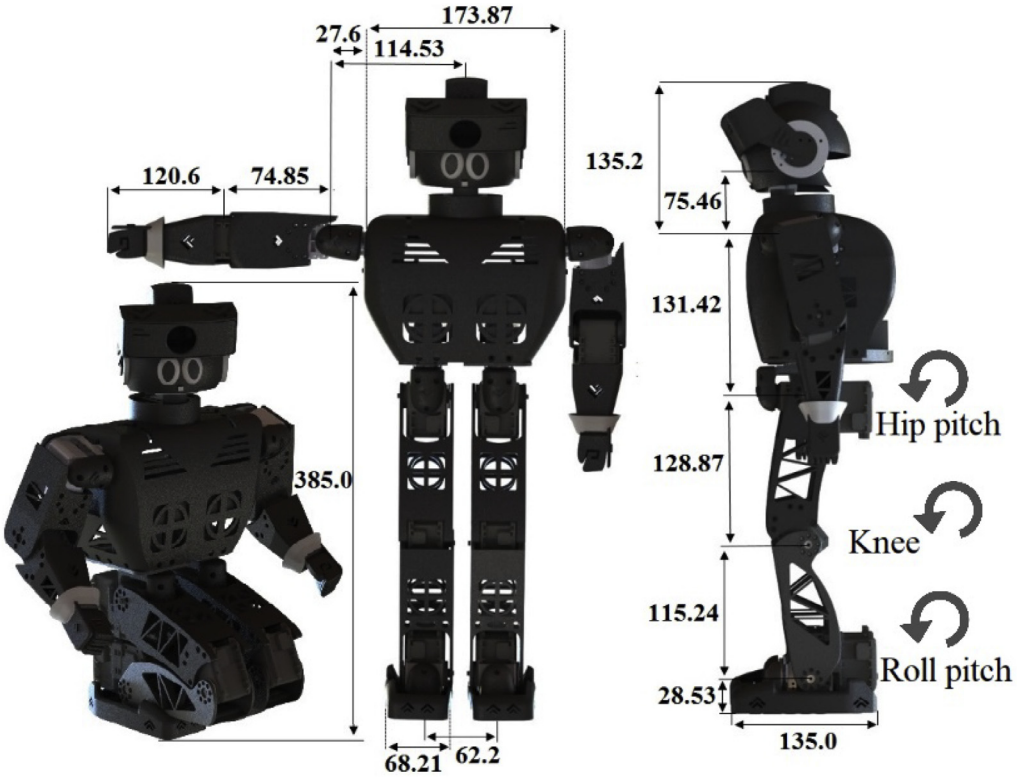

Fig. 1. Fully 3d printed ARC Humanoid Robot Platform along with robot's different parts sizes.

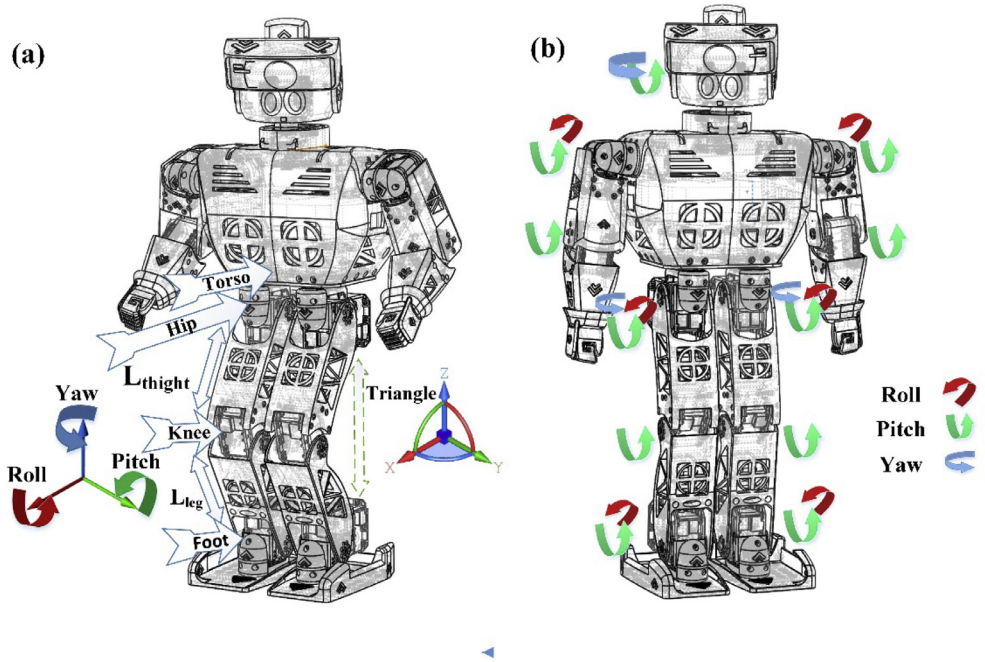

Fig. 2. (a) Visualization of coordinate frames used in the inverse kinematics, (b) ARC robot kinematic chains with 20 DOF joints and its initial stand.

regression algorithm which, according to [8], proved that computed mean absolute percentage error (MAPE) of the neural network model was significantly smaller than regression models.

It should be mentioned that among the existing dead reckoning based works on the humanoid robots, there are some approaches that still use magnetic sensors, others equip their robots with additional pressure sensors and some do not use IMU data as an influencing factor of odometry calculations. So in this paper, we provide LHOL method as a robust learning method to achieve an accurate odometry calculation.

3. ARC humanoid robot platform

ARC robot is designed and manufactured as a novel fully 3D printed Autonomous humanoid robot platform for Research and Competition

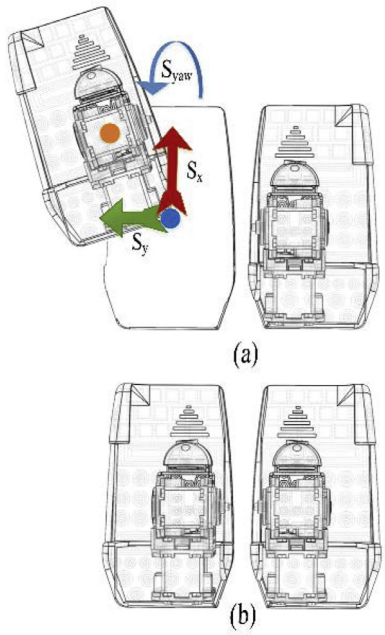

Fig. 3. (a) Illustration of the foot shifting at S_y,S_x and S_{yaw} within half-phases from a top-down view, (b) Illustration of the foot from an initial stand still at a top-down perspective.

purposes (Fig. 1). This platform is mechanically improved, lightweight, cheap, user-friendly for researchers, and equipped with wider camera field of view in comparison to kid-sized robots (90°FOV) of the same range. ARC robot platform has been built in our Humanoid Robots and Cognitive Technology research lab (HRCT) at the University of Tabriz in Iran.

Table 1
ARC Robot mechanical specifications.

Category	Specification	Value
Dimension	Height	54.0cm
	Weight	2.9Kg
DOF	Head	2 DOF
	Arm	2×3 DOF
	Leg	2×6 DOF
Actuator	6×AX-12A	TTL protocol, 11.1V
	14×MX-28	TTL protocol, 14.8V

The ARC robot's mechanical structure has been printed by a 3D printer in which the parts of the robot are designed as integrated as possible. In addition to robustness and simplicity, this integration requires less screwing and is cost-efficient. Table 1 illustrates the robot's mechanical specifications. Also, as illustrated in Table 2, ARC robot's main controller is Intel Core i5 6260 U which can cooperate with one 32-bit ARM Cortex-M3 processor at an open source CM9.04 board. Therefore, the processing system of ARC robot is more potent than other state-of-the-art platforms [15–17]. The actuators used in ARC robot are Dynamixel series which we can read its internal present load. Thus we don't equip the robot with additional sensors. Therefore, hip pitch, knee, and foot pitch actuators' internal present load can be used for measuring the load on the robot legs, which indicated in Fig. 1.

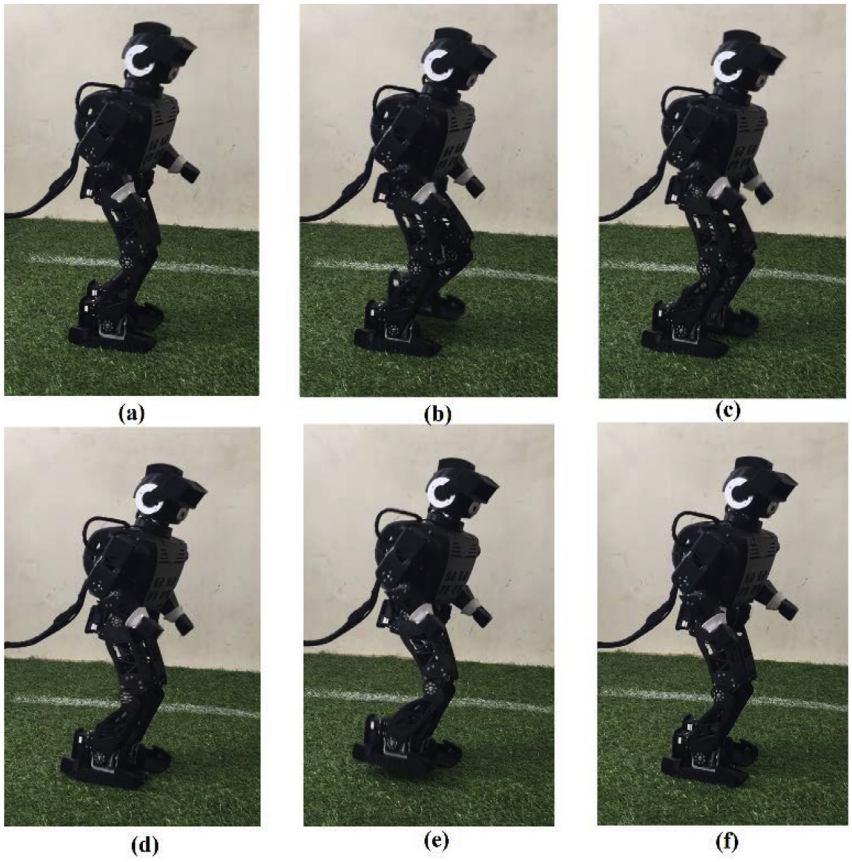

Fig. 4. Closed-loop walking implementation on the ARC robot, (a-f) left and right foot stepping respectively.

Table 2
ARC Robot electrical specifications.

Category	Specification	Value
Main controller	Model	Mini PC Intel NUC Kit NUC6I5SYK
	CPU	Intel Core i5 6260U
	Memory	8 GB DDR4 – 120 GB SSD M2
	Network	Ethernet, Wi-Fi, Bluetooth
	Other	4 × USB 3.0, HDMI, SDXC, 1x Mini DisplayPort
Sub controller	OpenCM9.04	32-bit ARM Cortex-M3
sensor	Gyroscope	3-Axis (Kalman filtered)
	Accelerometer	3-Axis (Kalman filtered)
	Compass	magnetic field (Kalman filtered)-Platform equipped but not used in this paper
	Camera	Logitech Pro Webcam - Wide-angle camera, 90°FOV

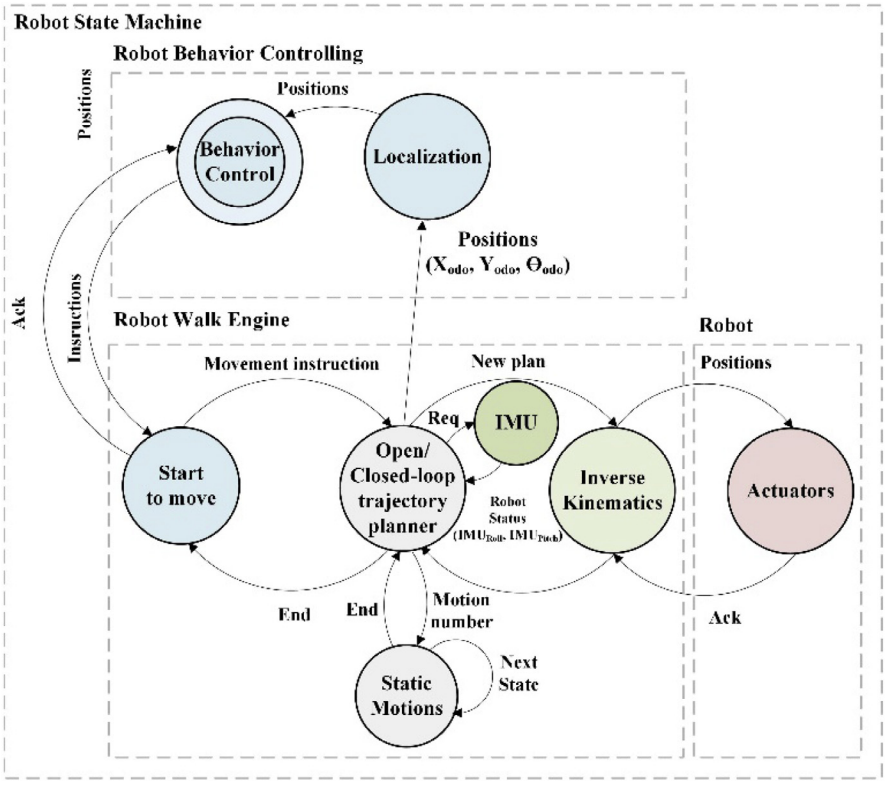

Fig. 5. State machine of the ARC robot platform including robot behavior controlling, robot walk engine, localization, and robot actuators controlling.

4. Lightweight humanoid robot odometric learning method (LHOL)

Dead reckoning (pure odometry) estimation is usually different from robot real movements and it is full of error at position and orientation estimations. It is mainly because of the robot's foot slipping or ground friction affections on overall dynamics. Hence in addition to the dead reckoning estimation, it is necessary to use robot's sensors environmental measuring data for achieving more accurate dead reckoning estimations. Therefore, for the odometry estimations, we proposed a powerful learning method based on the artificial neural networks (ANN) with three types of inputs. These three types of input data for learning are (i) dead reckoning calculations, (ii) filtered IMU data, and (iii) support leg present load data.

To calculate the dead reckoning as input data and to feed our Lightweight Humanoid robot Odometric Learning method (LHOL), it is necessary to implement inverse kinematics on the humanoid robot. Then, walk engine should be defined. Afterwards, the absolute Cartesian position of the robot over time can be calculated for the robot. Thus, in this section, first, we describe the inverse kinematics and walk engine to obtain the dead reckoning values as input data of learning network at LHOL method. Then, we describe learning with ANN in detail along with filtered IMU data and related to the support leg present load data.

4.1. ARC robot inverse kinematics

In this subsection, our implemented specific analytical solution to inverse kinematics in our provided novel fully 3D printed ARC robot is described briefly. The inverse kinematics for humanoid robots is provided in [18] and used in [19,20]. At the ARC robot, to solve the inverse kinematics problem analytically, the target of the feet was considered as homogenous transformation matrices. It means the matrices that contain the rotation and the translation of the foot in the coordinate system of the ARC robot torso.

In the same vein, all of the computations at inverse kinematics described for one leg of the ARC robot can be applied to the other legs as well. The position relative to the robot hip is a simple translation along the x-axis (see Fig. 2). The following Eq. (1) was used to calculate the transformation matrix of robot leg:

$$H(F) = Translate_{Roll}\left(\frac{L_d}{2}\right).T(F) \quad (1)$$

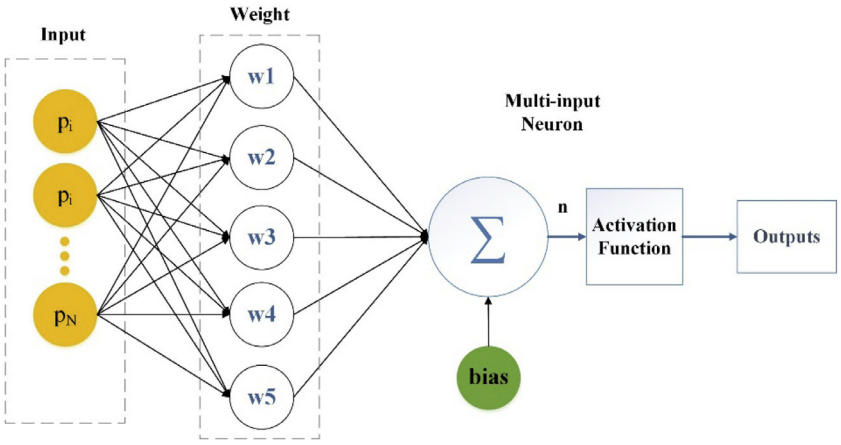

Fig. 6. Structure of the multi-input neuron, [30].

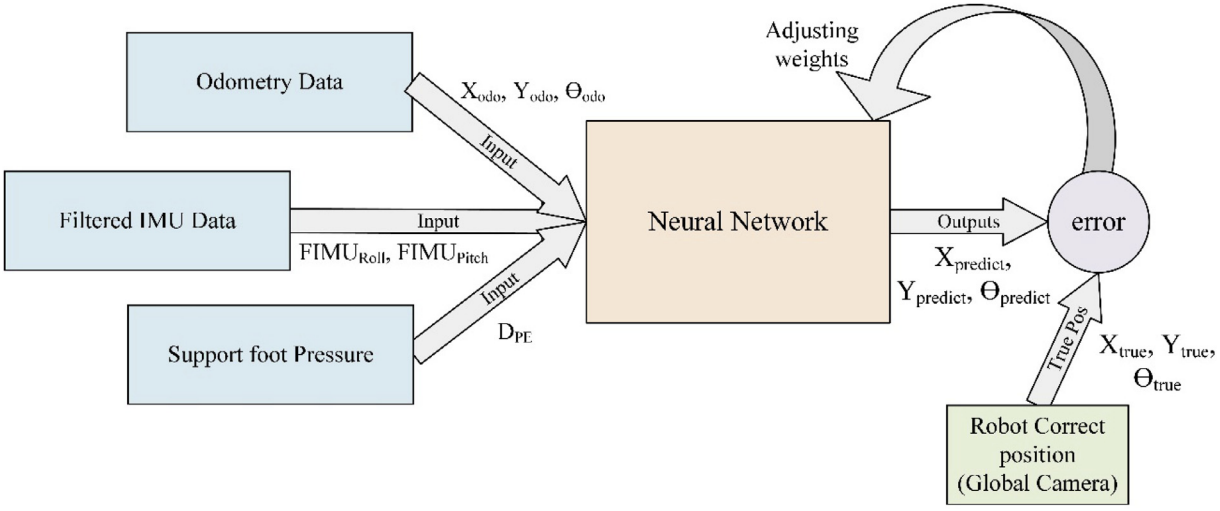

Fig. 7. The proposed method's architecture for training the odometry at ARC robot platform.

Where the $H(F)$ is a transformation matrix that transforms the point p_1 given as the coordinates of the hip to the same point p_2 in the coordinate system of robot's foot. The $T(F)$ is a transformation matrix that transforms the point p_1 given in the coordinate system of the torso to the same point p_2 in the coordinate system of the robot's foot joints.L_d is the distance between ARC humanoid robot' legs. According to the nature of the kinematic chain, this transformation is inverted, and the translational part of the transformation is solely determined by the Eq. (2).

$$F(H) = H(F)^{-1} \tag{2}$$

Therefore, the last three joints of the leg can be computed directly. Like most of the other humanoid robots such as NAO [21], and DARwIn-Op [22]. ARC robot limbs, knees and feet per leg form a triangle shape in which the edge equals the length of the translation vector of $F(H)$ as $L_{translation}$. Therefore, the angles of the triangle can be computed using the law of cosines. Accordingly, the hip pitch joint angle is calculated by the Eq. (3).

$$\delta_{hip\_pitch} = \arccos\left(\frac{{L_{thight}}^2 + {L_{leg}}^2 - {L_{translation}}^2}{L_{thight}.L_{thight}.L_{leg}}\right) \tag{3}$$

The δ_{hip_pitch} is the interior angle and the knee joint which is being stretched in the zero-position. Hence the resulting knee joint angle is computed by the Eq. (4).

$$\delta_{knee} = \pi - \arccos\left(\frac{{L_{thight}}^2 + {L_{leg}}^2 - {L_{translation}}^2}{L_{thight}.L_{thight}.L_{leg}}\right) \tag{4}$$

After calculating the knee joint, we computed the foot pitch joint of the robot by the Eq. (5), where x, y, z are the components and amount of the translation.

$$\delta_{foot\_pitch} = -\arctan\left(\frac{x}{z}\right) + \arccos\left(\frac{{L_{thight}}^2 + {L_{translation}}^2 - {L_{leg}}^2}{L_{thight}.L_{thight}.L_{leg}}\right) \tag{5}$$

Where δ_{foot_pitch} computes the angle opposite to the hip pitch. The foot roll and hip roll joints of the robot are calculated as δ_{foot_roll}, and δ_{hip_roll} respectively by the Eq. (6). So their joints can be derived from the translation vector using arctangent.

$$\delta_{foot\_roll} = \arctan\left(\frac{y}{z}\right), \delta_{hip-roll} = -\arctan\left(\frac{y}{z}\right) \tag{6}$$

Now all joints of the kinematic chain for one leg is computed. For the final implementation of the obtained angles for the joints of the robot at Radian, they must be converted to each of the final related servo motor positions by the Eq. (7).

$$\begin{aligned} {Act^i}_{pos} &= ({Act^i}_{res}/2) + {Act^i}_{Dir}\delta_{joint}(180/\pi)({Act^i}_{res}/360) \\ {Act^i}_{pos} &= ({Act^i}_{res}/2) + {Act^i}_{Dir}\delta_{joint}({Act^i}_{res}/2\pi) \end{aligned} \tag{7}$$

Where ${Act^i}_{res}$ is the resolution of the i-th related actuator to the obtained δ_{joint} of each joint (for instance servo Dynamixel MX series has

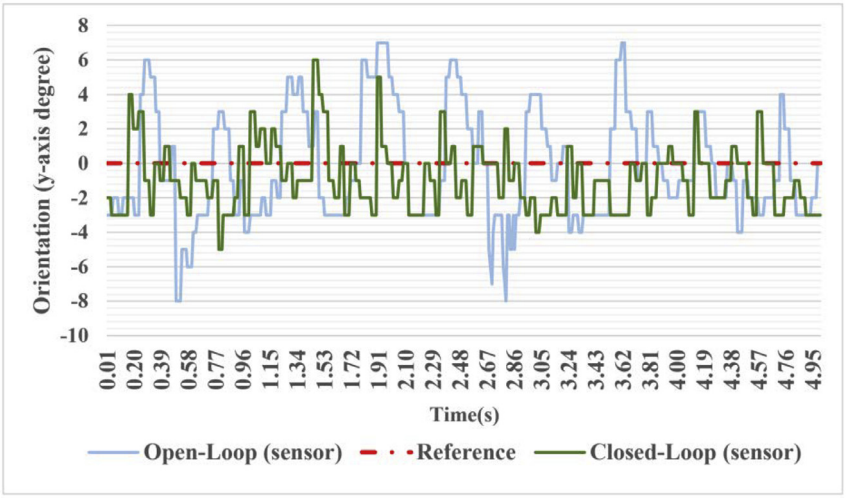

Fig. 8. ARC robot trunk orientation along the y-axis (Open-Loop and closed-loop with same gate frequency).

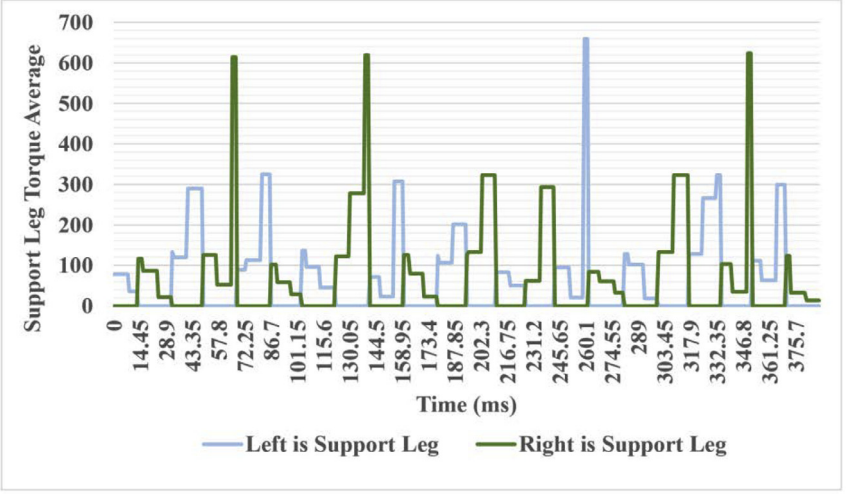

Fig. 9. ARC robot walking on artificial grass for 20 steps support leg average present load measurement (closed-loop with same gate frequency).

4096 resolution, so position 2048 is considered as its zero/initial position Fig. 2(b)). After computing joints positions for one leg, assuming that there can be kinematically different directions on both left and right legs, we determined directions of each actuators' movements at both legs by Act^{i}_{Dir}. So Act^{i}_{Dir} indicates the i-th servo motor direction to move as (± 1). Finally, the hip yaw joint of the robot can directly be calculated by angles with converting orientation of yaw angle to servomotor position by the Eq. (7). According to the descriptions in this section, we have six joints to realize a six DOFs per leg which is, as described above, solvable analytically. We name this section's calculations as (IK) by computing both legs' inverse kinematics with given x-y-z-axis as the components and amount of the translation per each leg.

4.2. Closed-loop walk engine

Walking control on the humanoid robot is one of the most challenging parts of it. The ability of Omnidirectional walk at humanoid robot enables it to accomplish walking in different directions and orientations with different speeds. To achieve this, two feet of the robot have to follow three-dimensional trajectories that define the actual steps. Omnidirectional walk at humanoid robots has been used in most of the state-of-the-art humanoid robots at different methods recently [23–26]. In the ARC robot, we determined the trajectories of the feet relative to its torso. The robot's legs positions are relative to the robot center of mass (CoM), and feet positions are relative to the trunk that moves the CoM to the desired direction and point. In this section, inverse kinematics (IK) was used for controlling the final positioning of robot's actuators as described in Section 4.1. Hence by giving the x–y–z-axis of each leg to IK, it generates joint angles and actuators' positions respectively.

The first step before the implementation of walk engine is to define the robot initial standing. By using servo motors in humanoid robots, and solid brackets the robot legs are not stretchable. Therefore, it is better to set the z-axis of the robot in sitting status through IK as shown in Fig. 2(a).

When the robot starts to walk, there are two separated half-phases. In each phase, one leg is the support leg, and the other one is the fly leg. We call both of the half-phases as one gate (0 and 2π). The step size and direction of both legs (support and fly) for movement at each half-phases will be generated through walk engine considering the robot's CoM. The CoM movement of the robot has to be performed along the x-axis and y-axis at walking directions. Robot CoM shifts at y-axis (S_y) to have one leg as the support leg, and another as the fly leg. Also, robot CoM shifts at x-axis (S_x) to move the robot forward. For the implementation of one gate, we followed Eqs. (8) and (9) as follows.

$$Leftleg\begin{cases} Ini_{left} + F_{fly}(S_x, S_y, S_{yaw}, Z_{fly}) & if(gate < \pi) \\ Ini_{left} + F_{support}(S_x, S_y, S_{yaw}) & elseif(\pi \le gate < 2\pi) \end{cases} \quad (8)$$

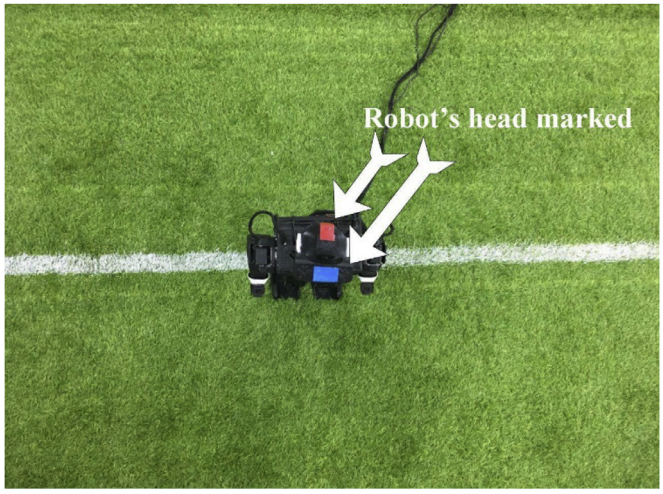

Fig. 10. Novel Fully 3D printed ARC robot on the artificial grass. The robot marked for tracking through the global camera at network training process.

$$Right\begin{cases} Ini_{right} + F_{fly}(S_x, S_y, S_{yaw}, Z_{fly}) & if(gate < \pi) \\ Ini_{right} + F_{\sup port}(S_x, S_y, S_{yaw}) & elseif(\pi \le gate < 2\pi) \end{cases} \quad (9)$$

Ini_{left}, and Ini_{right} are the feet's initial positions. S_y, and S_x are the current step sizes along the x-axis and y-axis respectively.S_{yaw} is the amount of rotation of leg along the yaw. Z_{fly} is the total offsets used for the amount of flying either the left or the right leg when one of them becomes the fly leg, (see Fig. 3). F_{fly} and $F_{\sup port}$ are functions in which parameterized trajectories are used for the foot flying and supporting respectively.

For calculation of F_{fly} and $F_{\sup port}$ as trajectories at fly and support foot, Eqs. (10) and (11) were used.

$$F_{fly} = \begin{cases} x = -\cos(gate).S_x \\ y = \cos(gate - \pi)S_y \\ z = \exp(((-gate)/Z_{freq})^2)Z_{fly} \\ yaw = \cos(gate - \pi)S_{yaw} \end{cases} \quad gate < \pi \quad (10)$$

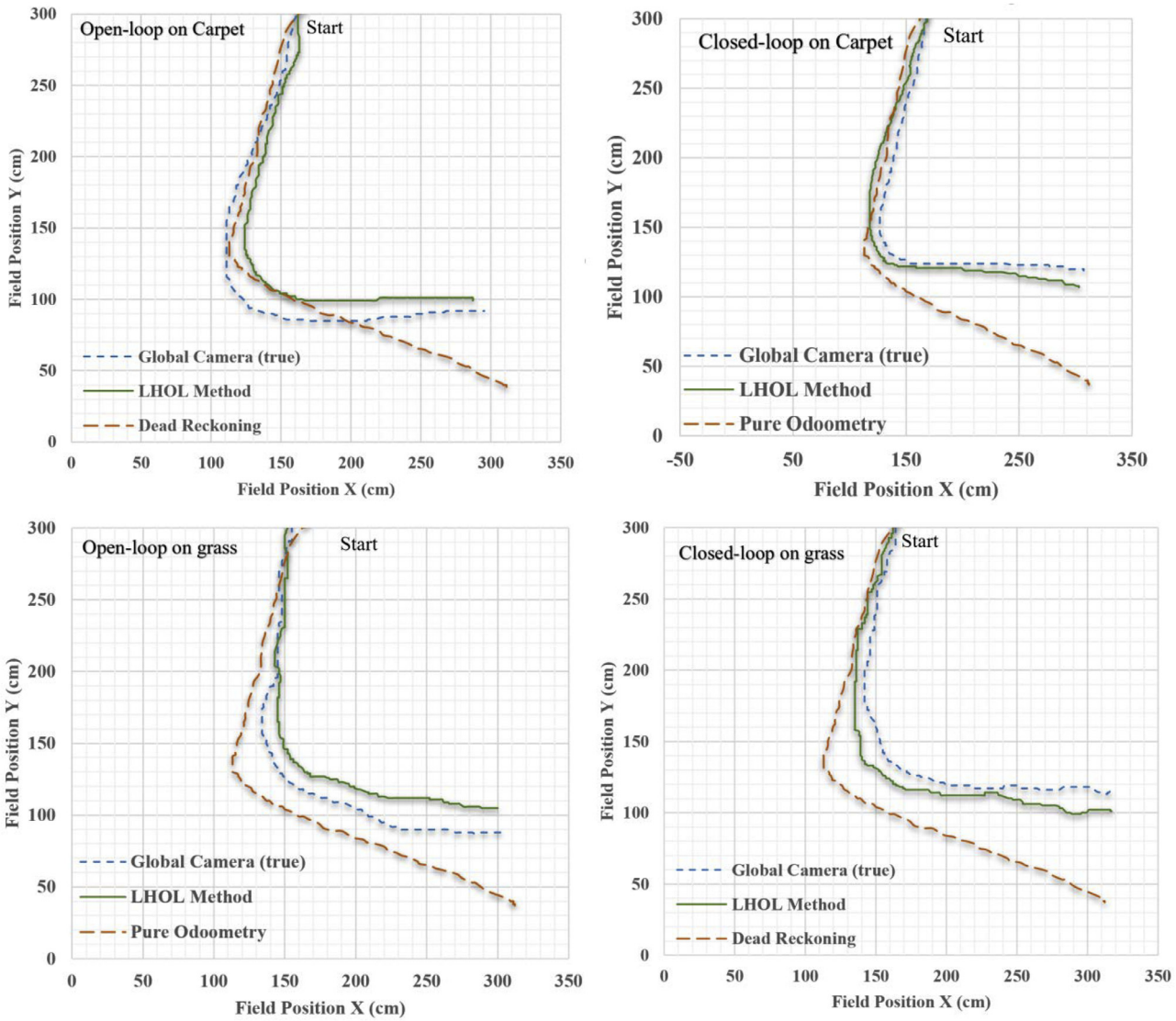

Fig. 11. The ARC robot platform is walking with both open-loop and close-looped walk engines on the artificial grass and carpet with the same walking instructions. Comparison between robot's true positions (tracked by a global camera), dead reckoning and LHOL method.

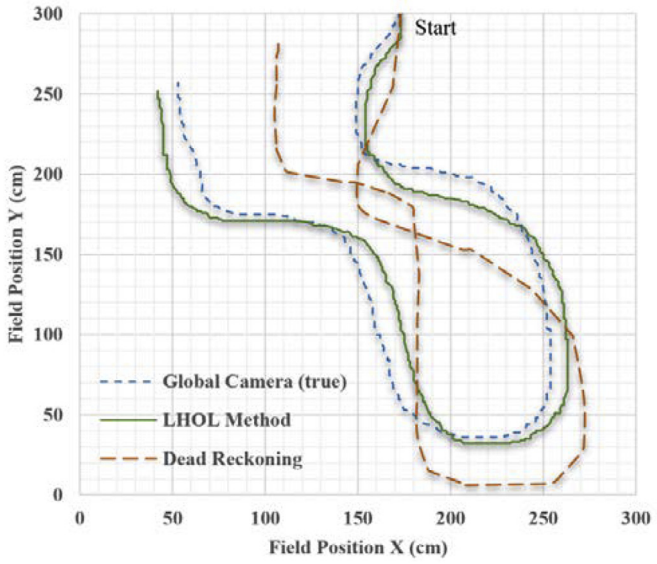

Fig. 12. Comparison between robot's true positions (tracked by a global camera), dead reckoning, and LHOL method.

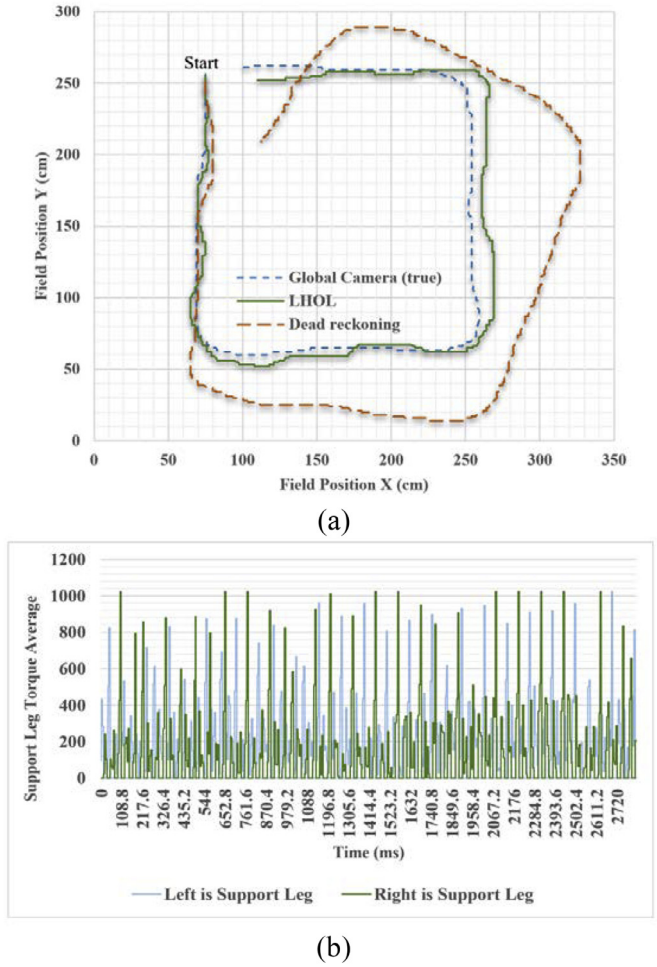

Fig. 13. (a) Comparison between robot's true positions (tracked by a global camera), dead reckoning, and LHOL method. This figure is obtained from robot's closed-loop walking on the artificial grass, (b) Support leg present load measurement along this experiment's walk duration.

$$F_{\text{sup }por} = \begin{cases} x = -\cos(gate - \pi)S_x \\ y = \sin(gate + \pi/T_{Shift})S_y \\ z = \exp(((-gate - \pi)/Z_{freq})^2)Z_{fly} \\ yaw = \cos(gate - \pi)S_{yaw} \end{cases} \quad \pi \leq gate < 2\pi \tag{11}$$

T_{Shift} is time-shifting movement at y-axis at support leg which causes changes at CoM on the support leg before starting to fly leg movements at x–z-axis and yaw direction. Z_{freq} determines the fly leg lifting and placing frequency. It is worth mentioning that CoM movement along the z-axis is useful, but it is ignored here to avoid more complexity.

Now, the robot can walk at open-loop trajectory by initializing the S_x, S_y, S_{yaw}, and Z_{fly} parameters. However, the importance of the closed-loop walking for dynamically balancing the robot is inevitable. Hence, for controlling the robot dynamically, we get feedback from Inertial Measurement Unit (IMU) with Kalman filtering the 3-axis of the accelerometer and 3-axis of the gyroscope (IMU_{rool}, IMU_{pitch}) [27]. The CoM of the robot balance has to be performed along the x-axis and y-axis in walking and should be corrected at each gate. Therefore, we change the walking trajectory to closed-loop trajectory by changing the calculations for support leg in the Eq. (12) as follows:

$$F_{\text{sup }por} = \begin{cases} x = -\cos(gate - \pi)S_x + P_{\text{PSG}}IMU_{pitch} \\ y = \sin(gate + \pi/T_{Shift})S_y + P_{\text{RSG}}IMU_{Roll} \\ z = \exp(((-gate - \pi)/Z_{freq})^2)Z_{fly} \\ yaw = \cos(gate - \pi)S_{yaw} \end{cases} \quad \pi \leq gate < 2\pi \tag{12}$$

P_{PSG} is the pitch stabilization gain which, at support foot, causes the movement of CoM in x-axis to balance the robot pitch direction. P_{RSG} is the roll stabilization gain. This gain just at support foot causes the change of CoM in y-axis to stabilize the robot roll direction.

In the walking trajectory, movement of the robot's arms helps to have more stability. To control robot's arms, we used a trajectory which moves the robot's arms just at pitch direction. The following trajectory in Eqs. (13) and (14) apply directly to the arm joints (described as IK at Section 2).

$$Arm_{fly\_leg\_side} = \begin{cases} Arm_{elbow} = \cos(gate)S_x + P_{\text{ASG}}IMU_{pitch} \\ Arm_{SholderPitch} = \cos(gate - \pi)S_x + P_{\text{ASG}}IMU_{pitch} \end{cases} \quad gate < \pi \tag{13}$$

$$Arm_{\text{sup port\_leg\_side}} = \begin{cases} Arm_{elbow} = \cos(gate - \pi)S_x + P_{\text{ASG}}IMU_{pitch} \\ Arm_{Sholder}Pitch = \cos(gate)S_x + P_{\text{ASG}}IMU_{pitch} \end{cases} \quad \pi \leq gate < 2\pi \tag{14}$$

P_{ASG} is the arm stabilization rate which enables the robot to have different stabilization rate at its arms. Fig. 4 illustrates the implementation of closed-loop walking on the ARC robot. Furthermore, for more clarification, the state-machine of the whole described inverse kinematics, walk engines, and their relation to localization and behavior control are shown in Fig. 5.

4.3. Dead reckoning calculation

The absolute Cartesian position of the robot over time can be calculated by taking into account the gait values at the described trajectory. According to the Section 4.2, we calculate robot dead reckoning by calculation of X_{odo}, Y_{odo} and θ_{odo} from Eq. (15).

$$Robot_{odometry} = \begin{cases} X_{odo} = |F_{fly\_leg}(x) - F_{\text{sup }por\_leg}(x)|S_x \\ Y_{odo} = |F_{fly\_leg}(y) - F_{\text{sup }por\_leg}(y)|S_y \\ \theta_{odo} = F_{fly\_leg}(S_{yaw}) + F_{\text{sup }por\_leg}(S_{yaw}) \end{cases} \quad gate = 2\pi \tag{15}$$

In Eq. (15) at the final step of each gate (at 2π), we can achieve the X_{odo}, Y_{odo} and θ_{odo} values which determine the amount of robot movement at x-axis, y-axis, and yaw orientation respectively. Therefore, we accomplish the dead reckoning calculation of the robot without considering mechanical and environmental errors.

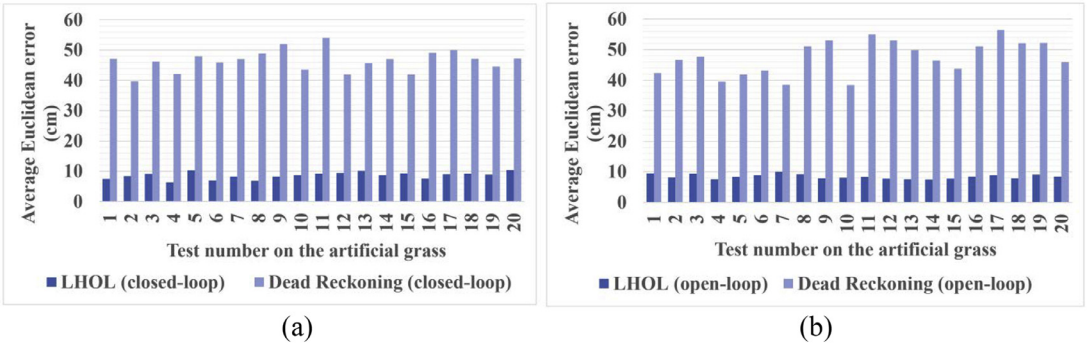

Fig. 14. Comparison between robot's true positions (tracked by a global camera), dead reckoning, and LHOL method at 20 different tests on the artificial grass. (a) Closed-loop wak engine, and (b) open-loop walk engine.

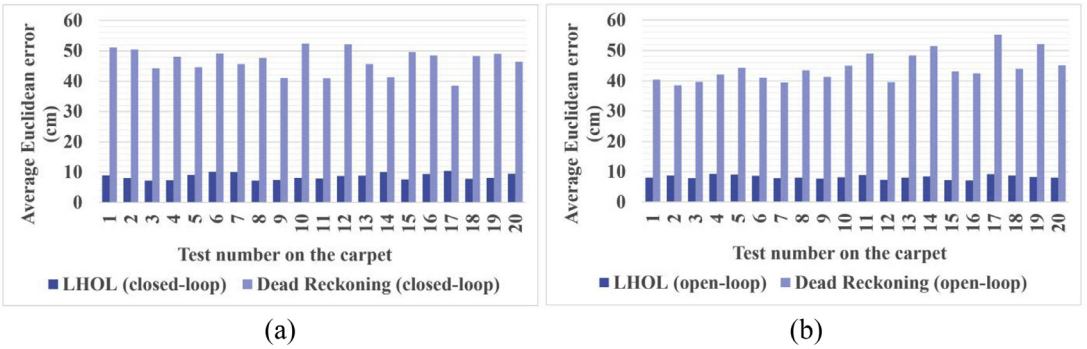

Fig. 15. Comparison between robot's true positions (tracked by a global camera), dead reckoning, and LHOL method at 20 different tests on the carpet. (a) closed-loop walk engine, and (b) open-loop walk engine.

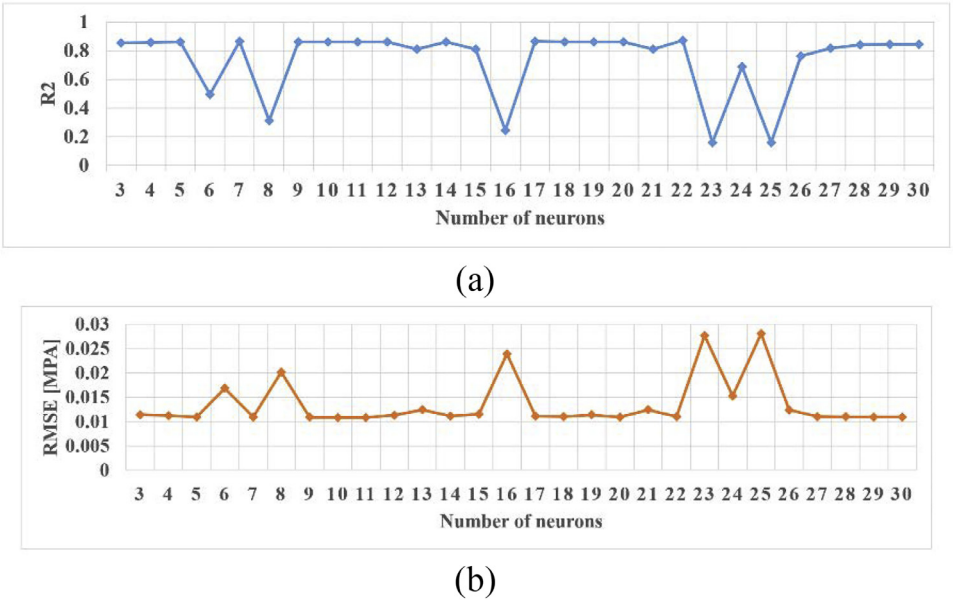

Fig. 16. Estimation performance of LHOL method with different number of hidden layer's neurons based on (a) R^2 and (b) RMSE.

4.4. Learning with ANN

There are many techniques for training a neural network to solve different problems [7]. The two useful and main methods employed by neural networks are known as supervised learning and unsupervised learning [28]. Supervised learning is almost the same as a human learning of new skills, by showing the network a series of the same examples. The most common supervised learning algorithm is known as backpropagation [29]. Hence it is considered as the primary learning method in this paper. So we provide a brief overview of the backpropagation training method as follows:

a) A set of samples for training the network is assembled. Each sample consists of inputs into the network and the corresponding solution which represents the desired (true) output from the network.
b) The input data is entered into the neural network via the input layer.

c) Each of the neurons in the network processes the input data, and the resultant values are steadily going through the network, at each layer, until some results are generated by the output layer.
d) The resulted output of the network is compared to expected output for that given particular input. These results have some errors which represent the differences between given input and expected output. All of the connection weights in the network are gradually adjusted considering errors values, working backward from the output layer, via the hidden layer, and to the input layer until the correct output is produced. Tuning the weights at the network properly trains it to provide the correct output for a particular input.

The training process is complete when the network produces the correct output for every input case with as light error value. The network's weights are saved in their trained states, and the network is then ready to be used in different real-time situations. New input data are presented to the network, and the trained network will determine the appropriate output on the basis of its training.

In this paper, a multilayer feed-forward neural network (FFNN) is chosen to estimate dead reckoning. This network is multi-layered perceptron (MLP) with a backpropagation training algorithm. A feed-forward neural network is composed of one input layer, one or more hidden layers, and one output layer. Since a neural network with one hidden layer has the capability to handle most of the complex functions, we consider the FFNN with one hidden layer. The basic element of a multilayer FFNN is neuron, which is a logical-mathematical model that seeks to simulate the behavior and functions of a biological neuron [30].

Fig. 6 shows the schematic structure of a neuron. As shown, typically, a neuron has more than one input. The elements in the input vector $p_{input} = [p_1, p_2, ..., p_i]$ are weighted by elements w_1, w_2, ..., w_i of the weight matrix W respectively.

Each neuron has a bias, which is summed with the weighted inputs to pass through an activation function, which can be expressed as below:

$$n = \sum_{j=1}^{N} w_j p_j + bias = W p + bias \tag{16}$$

The active function f with given n input generates the neuron output as $output = f(n)$. In this study, the log-sigmoid activation function is adopted which can be given by the following expression:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{17}$$

Thus, the multi-input FFNN in Fig. 6 implements the following equation.

$$output_2 = f_2\left(\sum_{i=1}^{S} w2_{1,i} f_1\left(\sum_{j=1}^{N} w1_{i,j} p_j + bias1_i\right) + bias2\right) \tag{18}$$

Where $output_2$ denotes the output of the overall networks. N is the number of networks' inputs, S is the number of neurons in the hidden layer. p_j indicates the jth input. $bias2$ is the bias of the neuron in the output layer and $bias1_i$ represents the bias of the ith neuron in the hidden layer. $w1_{i,j}$ represents the weight connecting the ith neuron of the hidden layer and the jth input, and $w2_{1,i}$ represents the weight of the output layer neuron connecting the ith source of the hidden layer. Also, f_1 and f_2 are the activation functions of the hidden layer and output layer, respectively.

Now, in order to train the established FFNN, the backpropagation algorithm can be utilized [31]. Backpropagation is a steepest descent algorithm, but the Levenberg-Marquardt algorithm is derived from Newton's method that was designed for minimizing functions which are sums of squares of nonlinear functions [32]. Thus, in this study Levenberg-Marquardt algorithm is used, (to see more about Levenberg-Marquardt algorithm refer to [33]). The detailed method and algorithm are introduced in the following subsection.

4.4.1. LHOL system architecture and learning details

The system architecture of the proposed method is illustrated in Fig. 7. In this figure, ANN is shown as a box which is trained with three types of input data including (i) dead reckoning calculations, (ii) filtered IMU data, and (iii) support leg present load simultaneously. Outputs of this network are compared to expected outputs for that given robot's true position and orientation. Hence, robot's true position, orientation, and consequently its momentum, is tracked using a global camera (just at training process). So the momentum X_{true}, Y_{true}, θ_{true} data obtained from the global camera were considered as the true values for feeding the learning algorithm. The output values of the ANN have some errors, which represent the differences between given input and expected output. Therefore, all of the weight and bias variables in the network are gradually adjusted with respect to errors values. To adjust these weight and bias variables, Levenberg–Marquardt method and the backpropagation algorithm are used to calculate the Jacobian matrix of the performance function with respect to the weight and bias variables. With updated weights and biases, the ANN further estimates the robot's momentum at the next time step. The training process is complete when the network produces the correct output for every input case with a slight error value.

Based on this figure, as learning network input data, we described (i) dead reckoning data calculation steps comprehensively at Section 4. However, both the (ii) filtered IMU data and (iii) support leg present load inputs will be described below.

Filtered IMU data: The filtered IMU impute data includes $FIMU_{Roll}$, $FIMU_{Pitch}$ values. These input data are obtained from robot's Kalman Filtered balancing sensor. Despite using Kalman Filtered data from IMU sensor; it is known that when a humanoid robot is walking (spatially at kid-sized robots), there is still ripple on data (Fig. 8). This ripple is because of the nature of humanoid robots when the robot is walking on uneven floors such as grass. Therefore, in the provided method, we define a specific filter on IMU data. This filter eliminates additional walking IMU data from the input. Eqs. (19) and (20) provide $FIMU_{Roll}$, $FIMU_{Pitch}$ filtering step respectively.

$$FIMU_{Roll} = \begin{cases} FIMU_{Roll} & if(FIMU_{Roll} > IMU_{RThreshold} \\ & ||FIMU_{Roll} < -IMU_{RThreshold}) \\ IMU_{RThreshold} & else \end{cases} \tag{19}$$

$$FIMU_{Pitch} = \begin{cases} FIMU_{Pitch} & if(FIMU_{Pitch} > IMU_{PThreshold} \\ & ||FIMU_{Pitch} < -IMU_{PThreshold}) \\ IMU_{PThreshold} & else \end{cases} \tag{20}$$

$IMU_{RThreshold}$ and $IMU_{PThreshold}$ are the thresholds for filtering the IMU data at roll and pitch directions. According to the equations, we just use the IMU data pick moments as learning network input. For more clarification, we illustrate ARC robot Open-loop walking on the artificial grass for ten steps at the same step frequency at Fig. 8. Its data was captured from Kalman Filtered IMU mounted on the platform. The threshold on the y-axis is defined as $IMU_{RThreshold} = 6$ for open-loop and $IMU_{RThreshold} = 2$ for closed-loop, which determine the uncertain unbalancing status of the robot's walking. This threshold depends on the humanoid robot's platform mechanical structure and implemented walk engine. Hence it is considered to be tuned by an expert user on walk engine for detecting and considering unmoral and unexpected instability of walk routine on the y-axis once.

Support leg present load: The Support leg present load data D_{PE} indicates the leg placement present load at stepping time periods by measuring support leg maximum load (Fig. 9). These data were collected from the three joints of robot support leg including foot pitch, knee, and hip pitch which are shown at Fig. 1 (this data can be read from internal

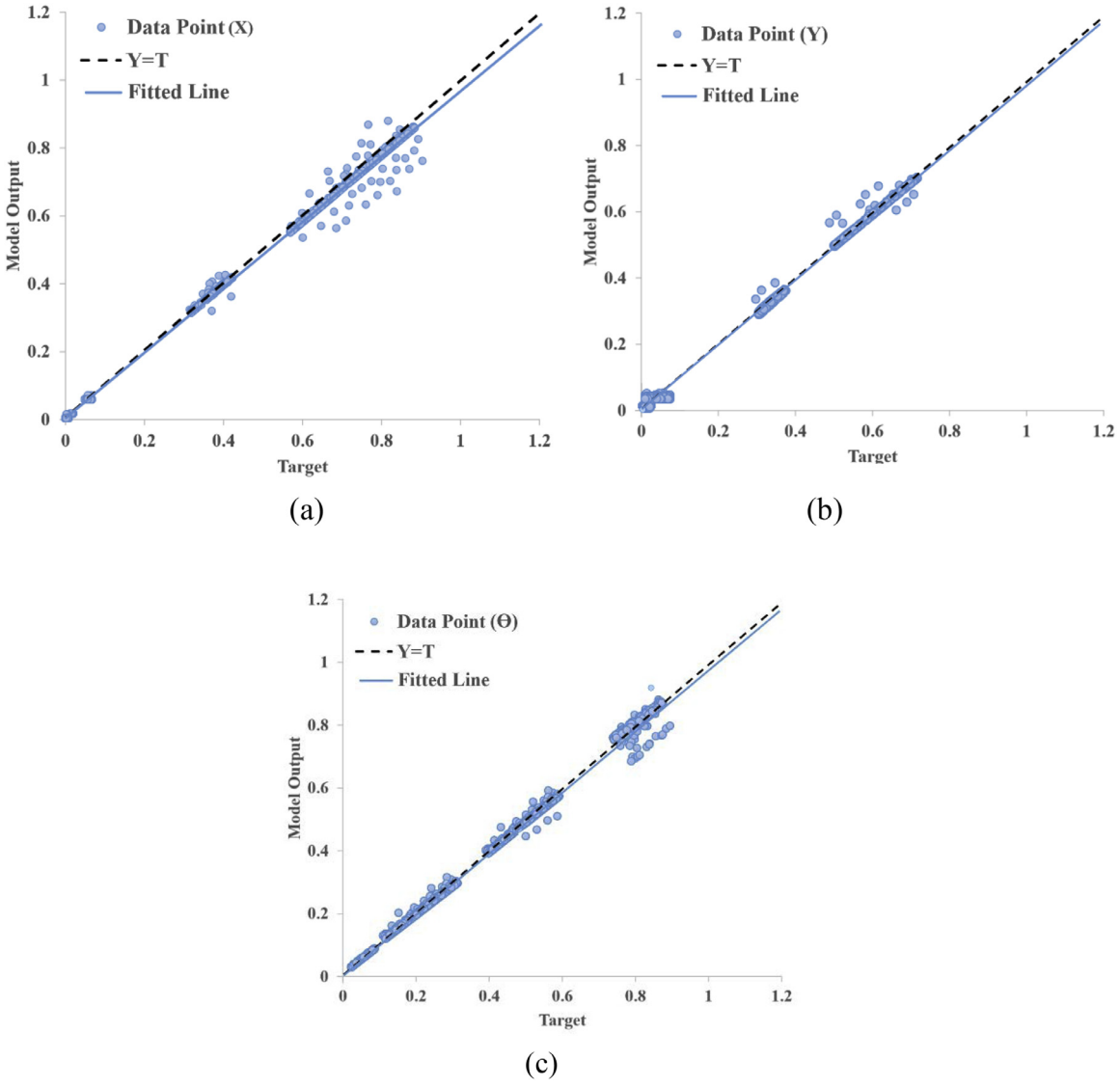

Fig. 17. Regression performance of the LHOL method's model with 5 neurons for three outputs for (a) $X_{predict}$, (b) $Y_{predict,}$ (c) $\Theta_{predict.}$

actuators' present load at servo motors, Table 1). It is clear that if the robot's support leg present load duration takes less or more than that of expected, it could mean that the robot's unexpected movement causes something like foot sleeping or unbalanced robot situations. Therefore, it should be considered as input data for learning network. We should note that it is possible that the sensed present load data for each actuator may not be very accurate. Therefore, when each leg becomes support, we take the average from mentioned three actuators' present load values. Furthermore, by training learning algorithm with comprehensive data, need for high accuracy in sensed loads data is eliminated.

5. Experimental results

In this section, to prove the accuracy of LHOL method first, we describe experimental setups. Then, we conducted and evaluated proposed LHOL. To this end, first, we show experiments with simple same walk instructions, second, we show experiments with complex omnidirectional walk instructions, third, we show the results of more tests on the proposed algorithm to discuss standard deviation. Then we analyze the utilized ANN learning algorithm and finally, the computational complexity of the proposed LHOL method is illustrated. It is worth mentioning that we do not analyze the walk controller robustness properties because trajectory tracking is not the focus of the paper.

5.1. Experimental setup

To evaluate proposed LHOL method, we implemented, conducted, and validated different experiments on the proposed ARC robot platform (Section 3). LHOL method uses ANN training algorithm with Levenberg-Marquardt backpropagation (as mentioned before it is used because of assuring fast convergence of the training error and being the widely used curve-fitting algorithm). The entire code was implemented in C sharp programming language and tested on the mounted main controller (mini-pc) inside the ARC robot platform. In this regard, standard Accord framework libraries were used to implement ANN which is compatible with .NET Framework, Windows Phone, Android, iOS, and Java [34]. Also, we described Inverse kinematics in Section 4.1 and a fully executable controlling software was developed for this platform.

Our experiments are conducted on the artificial grass with 3 cm blade height and carpet with 1 cm thickness. In the conducted experiments for both the closed-loop and open-loop walk engines, the trajectory in the straight line just was executed at the speed of 0.24 m/s, but in omnidirectional complex walk plans speed was between 0.1 m/s and 0.2 m/s. Note

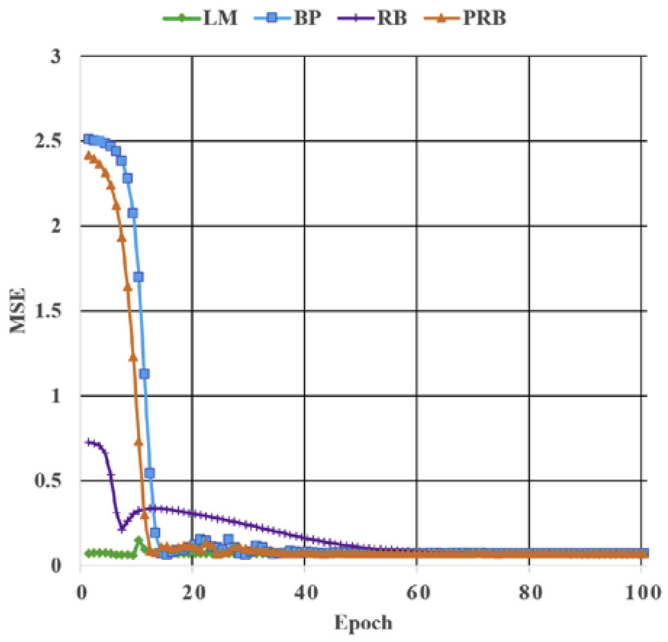

Fig. 18. Comparing of convergence rate of neural network training algorithms.

that the training process was based on three totall different walk speeds on each direction. In the training process of the network, the robot's position was tracked with a global camera, which we call it robot's true position (Fig. 10). So, LHOL method is compared to robot's true position and dead reckoning. The discussed average errors in the experiments were based on the robot's whole walking path Euclidean distance average error compared to the robot's true position tracked by a global camera simultaneously.

Error values were calculated by Euclidean distance from the robot's true position and related method's estimations in the walking path. The training process was done for both closed-loop and open-loop walk engines on both the carpet and artificial grass floors separately. The learning duration was 60 minutes for each of the closed-loop and open-loop walk engine on both types of floors.

5.2. Experimental results for simple walk instructions

Fig. 11 shows robot's walking with both open-loop and closed-looped walk engines on the artificial grass and carpet with the same walking instructions.

Fig. 11 represents four experiments conducted. Total path average error for LHOL and dead reckoning methods on the whole walking path in these experiments are 8.8 cm, and 23.3 cm respectively.

By looking at the results, it can be seen that the final robot's position error in dead reckoning is much worse than LHOL. For these experiments, we calculated LHOL method's improvement error rates by calculating average Euclidean distance error from the robot's true positions. Improvements were as follows: open-loop on the carpet = 46%, closed-loop on the carpet = 61%, open-loop on the artificial grass = 60%, and closed-loop on the carpet = 73%. Thus, the improvement average percentage error for these four experiments was 60%. These tests show the superiority of the LHOL method in decreasing error rate for both open-loop and close-looped walk engines on artificial grass and carpet.

5.3. Experimental results for complex walk instructions

To ensure accuracy of proposed method in more complex walk instructions and long distances, we conducted two other experiments, which were more complex with longer walking distance. In these experiments, the robot walked with two different walk instructions by closed-loop walk engine on both the carpet (Fig. 12), and grass (Fig. 13).

The results of the first complex experiment are illustrated in Fig. 12. In these results, the final robot's position Euclidean error with open-loop walk engine on the carpet for LHOL was 9.2 cm, whereas it was 59.5 cm for dead reckoning. Also, in this experiment total average walk path error for LHOL and dead reckoning were 9.7 cm and 42.6 cm respectively. In Fig. 13(a), the results of second complex experiment are illustrated. The results showed that the robot's final position Euclidean error for LHOL method was 9.8 cm, whereas it was 56.2 cm for dead reckoning, Also, average total walk path error for LHOL and dead reckoning were 8.3 cm, and 57.6 cm respectively as shown in Fig. 13.

By looking at Figs. 12 and 13(a) it becomes clear that the LHOL method decreased localization error in comparison to the dead reckoning dramatically. Consequently, in both of the experiments, the final robot's position error reduced from over 56 cm to less than 10 cm. Moreover, in both of these experiences, total average walks path error for the whole path reduced from over 42.6 cm to less than 9.8 cm. These results illustrate the superiority of the proposed LHOL method at complex walk instructions for both walking path average error and final robot's position error.

Furthermore, in Fig. 13(b), support leg present load measurement along walk duration has been shown. By looking at this present load measurement, we can see robot's walk instability and stability moments. As it is shown, the robot's right and left support leg sometimes sensed very large variant values between two continues steps, which illustrates instability moments in walk steps.

5.4. Experimental results for stability in complex walk instructions

In addition to previous experiments, to show the stability of LHOL method results at different experiments with the same walking instructions, we performed 20 tests per closed-loop and open-loop walk engines on both artificial grass (Fig. 14) and carpet (Fig. 15). These experiments were based on the walking instructions illustrated in Fig. 13. LHOL method and dead reckoning estimations errors were computed by the average Euclidean distance from the robot's true position at whole walk path.

Fig. 14 illustrates average error per 20 separate tests with robot open-loop and closed-loop walk engines on the artificial grass. The robot's average Euclidean error at both closed-loop and open-loop walk engines for LHOL method was 8.5 cm, whereas it was 46.9 cm for dead reckoning.

Moreover, to show the stability of the methods, we calculated the standard deviation for all of the experiments with closed-loop and open-loop walk engines. The standard deviation of error for LHOL and Dead Reckoning in closed-loop walk engine on the artificial grass were 1.11 cm and 3.5 cm as shown in Fig. 14(a). Also, the standard deviation of error and in open-loop walk engine were 0.72 cm and 5.95 cm, Fig. 14(b), respectively. Consequently, standard deviation achieved in these results shows the superiority of LHOL.

The average error per 20 separate tests with robot open-loop and closed-loop walk engines on the carpet is shown in Fig. 15. The robot's average Euclidean error at both closed-loop and open-loop walk engines for LHOL method was 8.4 cm, whereas it was 45.4 cm for dead reckoning.

In the same vein, to show the stability of the methods, we calculated the standard deviation for all experiments with closed-loop and open-loop walk engines. The standard deviation of error for LHOL and Dead Reckoning in closed-loop walk engine on the carpet was 1.07 cm and 3.96 cm (Fig. 15(a)) and in open-loop walk engine was 0.63 cm and 4.7 cm (Fig. 15(b)) respectively. So, the obtained standard deviation in second complex experiment shows the superiority of LHOL too. Consequently, experimental results in Figs. 14 and 15 prove the dominance of the proposed LHOL method in both average error and stability in different experiments, which makes it a robust method.

Table 3
The average error rate for open-loop and close-looped walk engines on the artificial grass and carpet for all experiments according to the robot positions tracked by a global camera, (Euclidean distance).

		LHOL method (Average error)	Dead reckoning (Average error)	Improvement (%)
Open-loop	Carpet	8.32 cm	43.23 cm	80.75
	Grass	8.45 cm	46.27 cm	81.73
Closed-loop	Carpet	8.62 cm	45.60 cm	81.09
	Grass	8.64 cm	45.92 cm	81.18
Total		**8.50cm**	**45.25cm**	**81.22**

Table 4
Comparison of the average error between the proposed method and state-of-the-art approaches on the humanoid robots. Results are the reported final localization average errors by each paper.

Approach		Method	Localization Detail	Required competitions	Required additional sensors	Average estimation error
Dead reckoning	1	Kumagai [6]	Gait Generator	low	Yes (compass)	50 cm
	2	Rouxel [14]	Foot pressure sensors + IMU + dead reckoning	low	Yes (pressure senor)	10.3 cm
	3	Oriolo [11]	Foot pressure sensors + IMU + joint encoders (built-in odometry error)	low	Yes (pressure senor)	9.53 cm
	4	**LHOL**	Dead reckoning + IMU + support leg present load	**low**	**No**	**8.50 cm**
Fusion-based	5	Oriolo [11]	VSLAM + foot pressure sensors + IMU + joint encoders	High	Yes (camera, pressure senor)	4.88 cm
	6	Kumagai [6]	SLAM + Gait Generator + Particle Filter	High	Yes (laser scanner + compass)	3.93 cm

In Table 3, average error rate for both open-loop and close-looped walk engines on the artificial grass and carpet for all previous experiments and their improvement's percentage have been shown.

Table 3 shows that average error value for LHOL method in all experiments is less than 9 cm, whereas in dead reckoning it is above 43 cm. The table also shows an improvement of 81.22% for the average error of LHOL method has in all experiments. However, robot's orientation estimation is an essential factor of odometry calculation. The robot's rotation's (orientation) estimation has a great importance at localization error. As mentioned earlier in the literature in this paper, we eliminated using compass sensor in the humanoid robot. Therefore, by accuracy in results of robot's position, the estimation of the robot's rotation precision was validated simultaneously too (estimations on θ_{odo}). Consequently, based on the robot's initial orientation, dependency on the compass sensor was eliminated in this paper. Results of the experiments indicated the superiority of the proposed LHOL method for different walk engines on different floor types, which makes it a robust method to be employed in a wide range of applications for the humanoid robots.

To show superiority of the proposed LHOL method against state-of-the-art approaches, we illustrated a comparison on the reported final localization average errors with related literature on the humanoid robots (Table 4). In this table related localization approaches are categorized as dead reckoning and fusion-based approaches, in which, as it is shown, proposed LHOL method achieved the lowest error in average estimations while it does not use additional sensors. As it can be seen, fusion-based approaches which combine VO and dead reckoning archived better results than all dead reckoning approaches. It is obvious that the main reason is that in VO, the proposed algorithm can eliminate accumulated errors in time duration easily. However, proposed LHOL method has best results on the dead reckoning approaches category. So as stated before, LHOL method can be used solely or in combination with other VO approaches based on the applications requirements.

Table 5
Key parameter of FFNN on the LHOL method.

Parameter	Value
Maximum number of epochs to train	500
Performance goal	0
Test hidden layer's neurons numbers	3–30
Maximum time to train in seconds	Infinite

5.5. Evaluation the estimations of the ANN-based dead reckoning

In addition to the emperical evaluation of robot's dead reckoning estimations in previous sections, we modulate and evaluate the learning model performance in this section seperately. First, some key parameter of FFNN on the LHOL method are illustrated in Table 5.

In this section, the K-fold cross-validation approach is adopted [35]. In the K-fold cross-validation approach, among the K folds divided, (K-1) ones are selected to train the model, and the rest is utilized as testing data. Thus, the overall recorded data from robot's tests are divided into two sets, and the final evaluation of the model performance is carried out based on the K test results, which in this work, the value of K is set as 5.

To quantitatively evaluate the estimation performance of the learning method, two commonly used indices including, root-mean-square-error (RMSE) or root-mean-square deviation (RMSD), and coefficient of determination R^2 are adopted. The definitions of the RMSE and R^2 are presented as follows. Suppose the reference data is $True = \{t_1, t_2, ..t_N\}$, and the predicted value is $Y = \{y_1, y_2, ..y_N\}$. Then R^2 can be calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (t_i - y_i)^2}{\sum_{i=1}^{N} (t_i - \overline{True})^2} \tag{21}$$

where $\sum_{i=1}^{N} (t_i - y_i)^2$ is the residual sum of square, $\sum_{i=1}^{N} (t_i - \overline{True})^2$ is the total sum of square, and $\overline{True}$ is the mean value of the reference data. Also, the RMSE can be obtained by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (t_i - y_i)^2}{N}} \tag{22}$$

Firstly, the impact of the neuron number on the LHOL method estimation performance is analyzed. Considering the complexity of the problem, which with filtered inputs tried to reduce the estimation performance, is tested under a different number of hidden layer's neurons ranging from 3 to 30. According to Fig. 16, as the number of hidden layer's neurons changes, the estimation accuracy of the LHOL method varies slightly. The best prediction performance is produced by LHOL

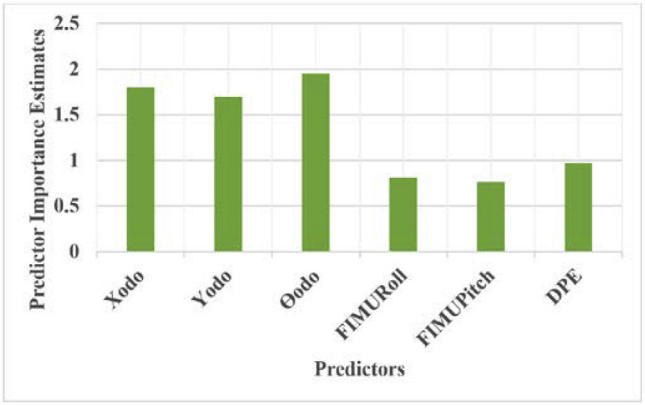

Fig. 19. The predictor importance estimation results.

method with the number of neurons at 5 and 20. We chose 5 number of neurons because of its fast learning process.

Then, the linear regression performance of the trained model for each three output neurons is investigated. Based on the linear regression result shown in Fig. 17, the test regression result are $R_X = 0.9576$, $R_Y = 0.9848$, $R_\theta = 0.979$, which indicate that the LHOL method model with 5 neurons can accurately estimate the robots movement momentum for three outputs ($X_{predict}$, $Y_{predict}$, $\theta_{predict}$) through selected features respectively. Note that a network with few hidden nodes will have limited learning capabilities, while an excess of hidden nodes will lead to overfitting or generalization loss [36]. Hence, based on the learning network structure and obtained results we are not faced with the overfitting and underfitting problems on the proposed method.

To analyze the different learning algorithms for training algorithms (ANN in Fig. 18), we compared Levenberg-Marquardt Learning (LM) with three other learning algorithms including Back Propagation Learning (BP), Resilient Backpropagation Learning (RB), and Parallel Resilient Backpropagation Learning (PRB). In this figure after training the network at each epoch, we evaluated the learning algorithms with test data to show Mean Square Error (MSE). As shown in Fig. 18, the LM is reduced with the most speed than other algorithms based on the MSE at different epochs. This shows LM's faster convergence and superiority for our problem in comparison to other learning approaches.

In this research to show importance of the analysis of the selected features, the utilized feature variables were further investigated by analyzing the importance of predictors [37]. In Fig. 19 we illustrated the estimation results of the predictor importance. In this figure, a larger value of the predictor importance indicates that the feature variable has a greater effect on the model output.

Based on the results in Fig. 19, the important features in the models are odometry features (X_{odo}, Y_{odo}, θ_{odo}) in which the yaw orientation feature θ_{odo} (robot's rotation odometry feature) was the most important one.

5.6. LHOL computational complexity

The computational complexity of the provided LHOL method is illustrated in Table 6. As it is shown in the table, the average computational time for odometry is very low. Main reasons for this low time are: first, computing odometry with the walk engine loop simultaneously, second, not using high computational VO approaches. Hence, LHOL is a lightweight approach which can be used in humanoid robots' real-time localization without additional computational power requirements.

6. Conclusion

Localization is one of the most important parts of robot behavior control. Vision or laser scanner based approaches for localization require high computational power for analyzing visual information. Therefore, these approaches are not suitable for all of the lightweight robots' weak embedded systems. On the other hand, magnetic sensors have instability problems. Moreover, adding additional pressure sensors increase the cost of the kid-sized humanoid robots. Because of the complex mechanical system of humanoid robots and due to the presence of many sources of uncertainty and inaccuracy in motion execution such as foot slippage, the accurate calculation of dead reckoning (pure odometry) is hard. Therefore, we presented a lightweight and robust humanoid robot odometric learning method (LHOL) for localization at the humanoid robots. This method uses dead reckoning calculations, IMU data, and actuators' internal present load in walking steps as learning network input. Thus, this method does not use vision, magnetic, additional pressure, and laser scanners. So, we have demonstrated the possibility of providing a machine learning method to significantly enhance the dead reckoning as odometry with lightweight computational requirements. The LHOL method proved high accuracy (total average error at all experiments was less than 9 cm) on a novel fully 3D printed kid-sized humanoid robot platform (ARC). Experiments were obtained with both open-loop and closed-loop walk engines at differently covered floors (carpet and artificial grass). The obtained results showed that this powerful learning method can be used as a localization method for humanoid robots. Also, LHOL method can be used alone or in combination) with other localization approaches (to decrease accumulated errors such as monocular vision localization depending on its application.

Table 6
Average computing time on robot's main controller CPU.

	Specification	Time
Main control loop	100 Hz	~5.9 ms
Logging	IMU Sensor	0.3 ms
	Actuators	0.7 ms
	Dead reckoning	0.2 ms
Walk engine	Open-loop	0.5 ms
	Closed-loop	0.6 ms
	Odometry Prediction	3.9 ms

References

[1] Lee K, Chung W, Yoo K. Kinematic parameter calibration of a car-like mobile robot to improve odometry accuracy. Mechatronics 2010;20:582–95.
[2] Tavakoli M, Lopes P, Sgrigna L, Viegas C. Motion control of an omnidirectional climbing robot based on dead reckoning method. Mechatronics 2015;30:94–106.
[3] Wang D, Liang H, Zhu H, Zhang S. A bionic camera-based polarization navigation sensor. sensors 2014;14:13006–23.
[4] Aqel MOA, Marhaban MH, Saripan MI, Ismail NB. Review of visual odometry: types, approaches, challenges, and applications. Springerplus 2016;5.
[5] Baltes J, Gerndt R, McGill S, Sadeghnejad S. RoboCup soccer humanoid league rules and setup. International RoboCup Federation; 2016.
[6] Kumagai I, Ueda R, Sugai F, Nozawa S, Kakiuchi Y, Okada K, Inaba M. Achievement of localization system for humanoid robots with virtual horizontal scan relative to improved odometry fusing internal sensors and visual information. IEEE/RSJ international conference on intelligent robots and systems (IROS) Daejeon , Korea. IEEE; 2016.
[7] Gray AR, MacDonell SG. A comparison of techniques for developing predective models of software metrics. Inf Softw Technol 1997;39:425–37.
[8] Heiat A. Comparison of artificial neural network and regression models for estimating software development effort. Inf Softw Technol 2002;44:911–22.
[9] Wang K, Liu Y-H, Li L. A simple and parallel algorithm for real-time robot localization by fusing monocular vision and odometry/AHRS sensors. IEEE/ASME Trans Mechatron 2014;19:1447–57.
[10] Oriolo G, Paolillo A, Rosa L, Vendittelli M. Vision-based odometric localization for humanoids using a kinematic EKF. IEEE-RAS international conference on humanoid robots. Japan: IEEE, Business Innovation Center Osaka; 2012.
[11] Oriolo G, Paolillo A, Rosa L, Vendittelli1 M. Humanoid odometric localization integrating kinematic, inertial and visual information. Auton Rob 2015;40:867–79.
[12] Martínez PA, Lin X, Castelán M, Casas J, Arechavaleta G. A closed-loop approach for tracking a humanoid robot using particle filtering and depth data. Intel Serv Rob 2017:1–16.
[13] Schmitz A, Missura M, Behnke S. Learning footstep prediction from motion capture. In: Lecture notes in computer science. Berlin, Heidelberg: Springer; 2010. p. 97–108.

[14] Rouxel Q, Passault G, Hofer L, N'Guyen S, Ly O. Learning the odometry on a small humanoid robot. IEEE international conference on robotics and automation (ICRA) Stockholm, Sweden; 2016.
[15] Ha I, Tamura Y, Asama H. Development of open platform humanoid robot DARwIn-OP. Adv Rob 2013;27:223–32.
[16] Allgeuer P, Farazi H, Ficht G, Schreiber M, Behnke S. The igus humanoid open platform-a child-sized 3D printed open-source robot for research. Ku¨nstl Intell 2016;30:315–19.
[17] Gouaillier D, Hugel V, Blazevic P, Kilner C, Monceaux Jeo, Lafourcade P, Marnier B, Serre J, Maisonnier B. Mechatronic design of NAO humanoid. International conference on robotics and automation Kobe, Japan. IEEE; 2009.
[18] Tolani D, Goswami A, Badler NI. Real-time inverse kinematics techniques for anthropomorphic limbs. Graphical Models 2000;62:353–88.
[19] Graf C, H¨artl A, R¨ofer T, Laue T. A robust closed-loop gait for the standard platform league humanoid. In: Intl. conf. on humanoid robots; 2009. p. 30–7.
[20] Kofinas N, Orfanoudakis E, Lagoudakis MG. Complete analytical forward and inverse kinematics for the NAO humanoid robot. J Intell Rob Syst 2015;77:251–64.
[21] Gouaillier D, Hugel V, Blazevic P, Kilner C, Monceaux Jeo, Lafourcade P, Marnier B, Serre J, Maisonnier B. Mechatronic design of NAO humanoid, robotics and automation. Kobe, Japan: IEEE; 2009.
[22] Ha I, Asama YTaH. Development of open platform humanoid robot DARwIn-OP. Adv Rob 2013;27:223–32.
[23] Lee K, Ryoo Y-J. Omni-directional walking pattern generator for child-sized humanoid robot, CHARLES2. Int J Humanoid Rob 2017;14:1–16.
[24] Snafii N, Abdolmaleki A, Lau N, Reis LP. Development of an omnidirectional walk engine for soccer humanoid robots. Int J Adv Rob Syst 2015;12:1–14.
[25] Seekircher A, Visser U. An adaptive LIPM-based dynamic walk using model parameter optimization on humanoid robots. KI Künstliche Intelligenz 2016;30:233–44.
[26] Gouaillier D, Collette C, Kilner C. Omni-directional closed-loop walk for NAO. International conference on humanoid robots Nashville, TN, USA. IEEE; 2010.
[27] Zhu R, Sun D, Zhou Z, Wang D. A linear fusion algorithm for attitude determination using low cost MEMS-based sensors. Measurement 2007;40:322–8.
[28] Karayiannis NB, Mi GW. Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques. IEEE Trans Neural Netw 1997;8:1492–506.
[29] Aydogmus Z, Aydogmus O. A comparison of artificial neural network and extended Kalman filter based sensorless speed estimation. Measurement 2015;63:152–8.
[30] Demuth HB, Beale MH, De Jess O, Hagan MT. Neural network design. Martin Hagan; 2014.
[31] Soualhi A, Makdessi M, German R, Echeverría FR, Razik H, Sari A, Venet P, Clerc G. Heath monitoring of capacitors and supercapacitors using the neo-fuzzy neural approach. IEEE Trans Ind Inf 2018;14:24–34.
[32] Nasrabadi NM. Pattern recognition and machine learning. J Electron Imaging 2007;16:049901.
[33] Dreyfus G. Neural networks: methodology and applications. Springer Science & Business Media; 2005.
[34] Accord.NET framework, http://accord-framework.net/.
[35] Refaeilzadeh P, Tang L, Liu H. Cross-validation, encyclopedia of database systems. Springer; 2009. p. 532–8.
[36] Cortez P, Rio M, Rocha M, Sousa P. Internet traffic forecasting using neural networks, neural networks, 2006. IJCNN'06. In: International joint conference on. IEEE; 2006. p. 2635–42.
[37] Budescu DV. Dominance analysis: a new approach to the problem of relative importance of predictors in multiple regression. Psychol Bull 1993;114:542.

Saeed Saeedvand received his B.S. degree in Computer Engineering in 2012. He received his M.S. degree in Computer Engineering from the University of Tabriz in 2014. Currently, he is a Ph.D. candidate in Information Technology Engineering at the University of Tabriz. He has been a lecturer at the University of Tabriz since 2014. He has been worked on the humanoid adult-size, and kid-size robots since 2009, and he obtained two 3rd places at RoboCup 2016 and 2017 adult-size humanoid robot competitions and three 1st places at IranOpen humanoid robot competitions. He is a member of the technical committee of IranOpen robotic competitions since 2016. His research interest includes artificial intelligence, robotics, machine learning, and computer vision.

Hadi S. Aghdasi received his B.S. degree in computer engineering in 2006 from Sadjad University of Technology, Mashhad, Iran and received his M.S. and Ph.D. degrees in computer engineering from Shahid Beheshti University, Tehran, Iran, in 2008 and 2013, respectively. He has been an assistant professor of Computer Engineering department at University of Tabriz, Tabriz, Iran since 2013. His current researches focus on Humanoid Robots and Intelligent Methods in Surveillance Systems and Cognitive Technology. Also, he works on Wireless Traditional and Visual Sensor Networks (Routing, Clustering, Coverage, and Visual Information Transmission). He is a director of the both Humanoid Robots and Cognitive Technology (HRCT) and Wireless Ad hoc and Sensor networks (WASL) research laboratories in University of Tabriz, Tabriz, Iran.

Jacky Baltes received his Ph.D. degree in 1996 from the University of Calgary in Artificial Intelligence. From 1996 to 2002, he worked as a senior lecturer at the University of Auckland in Auckland, New Zealand. Since 2002, he has been a professor in the Department of Computer Science at the University of Manitoba in Winnipeg, Manitoba. He has worked as an Outstanding Professor and head of Educational Robotics Center at the National Taiwan Normal University, Taiwan, since 2016. He has also been a vice president of the FIRA robotic soccer association, chair of the HuroCup competition, and a member of the RoboCup executive committee. His research interests are intelligent robotics, artificial intelligence, machine learning, and computer vision.