Maneuverable Autonomy of a Six-legged Walking Robot: Design and Implementation using Deep Neural Networks and Hexapod Locomotion

Hiep Xuan Huynh[1*], Nghia Duong-Trung[2], Tran Nam Quoc Nguyen[3], Bao Hoai Le[4], Tam Hung Le[5]
Can Tho University, Can Tho city, Vietnam[1,3,4,5]
Technische Universit¨at Berlin, Berlin, Germany[2]
Corresponding Author*

***Abstract*—Automatically real-time synthesizing behaviors for a six-legged walking robot pose several exciting challenges, which can be categorized into mechanics design, control software, and the combination of both. Due to the complexity of controlling and automation, numerous studies choose to gear their attention to a specific aspect of the whole challenge by either proposing valid and low-power assumption of mechanical parts or implementing software solutions upon sensorial capabilities and camera. Therefore, a complete solution associating both mechanical moving parts, hardware components, and software encouraging generalization should be adequately addressed. The architecture proposed in this article orchestrates (i) interlocutor face detection and recognition utilizing ensemble learning and convolutional neural networks, (ii) maneuverable automation of six-legged robot via hexapod locomotion, and (iii) deployment on a Raspberry Pi, that has not been previously reported in the literature. Not satisfying there, the authors even develop one step further by enabling real-time operation. We believe that our contributions ignite multi-research disciplines ranging from IoT, computer vision, machine learning, and robot autonomy.**

***Keywords*—Six-legged walking robot; hexapod locomotion; Raspberry Pi; deep neural networks**

I. Introduction and Motivation

Controlling a multi-legged robot receives a vast research endeavor in the literature. At first, researchers scrutinized the advantages of a legged robot over a wheeled one by investigating the degrees of freedom, mechanical structure, gait generation, and body stabilization on difficult terrain [1]. Then a scheme of the locomotion control approach by mimicking the biological movement of legs and basic motion pattern was investigated. The simulation and modeling were developed on either the eight-legged or six-legged robot. The feedback taken from the environment was collected from motor sensors [2]. By mimicking the physical structure of legged animals, engineers can implement a more stable and faster walking robot. A six-legged walking robot whose kinetic chain of legs based on the biomechanics of the cockroach was prototyped by [3]. The motion mechanisms were controlled by a PLC controller, and the two-degree-of-freedom robot only moved in a straight direction. Another work researched dynamic forward legged locomotion toward a static object using a single monolithic controller. Two virtual quadrupedal robots were used [4]. Another two papers have done intensive experiments and modeling of the robot stability and energy consumption analysis [5], [6]. Many other research articles have strengthened the aspect of sensor-based feedback, and mechanics can be found in [7], [8], [9].

Computer vision on Raspberry Pi is overwhelming on its own field [10], [11], [12]. The tasks can be narrowed to object detection and recognition, home automation, and surveillance, combining camera and software libraries [13], [14]. The idea of real-time face detection deployed on a telepresence robot was proposed in [15], where two Raspberry Pi boards were utilized to achieve real-time detection. They developed a technique to enable real-time detection on a Raspberry Pi and using the result to control the pan and tilt unit of a telepresence robot. The user can either robot's movement by turning or rotating the smartphone or set the robot to follow the face of someone who is at the robot place.

This article aims to bring the two separate research directions into one choir where robot automation, IoT, computer vision, and machine learning operate several complex tasks in real-time. Many contributions of owner work are follows. First, we propose a prototype and mechanical structure of a six-legged working robot. Second, we propose an architecture of convolutional neural networks that work very effectively regarding highly accurate face recognition and be able to deploy on a Raspberry Pi. Third, we implement a complete robot automation solution.

The article is organized as follows. Section (II) briefly presents the background on Haar-like features, boosting technique in machine learning, the cascade of weak classifiers to form a strong one, and convolutional neural networks. These materials help readers understand the underlying techniques that address the tasks of face detection and recognition. Section (III) summarizes several main hardware components used in the experimental robot. Many software libraries, e.g., especially related to convolutional neural networks, are introduced in Section (IV). Our main contributions of this research article are presented in Section (V), where the authors describe the system architecture and its three sub modules, e.g., face detection, owner recognition, and kinematic automation. Section (VI) present the experiments and experimental remarks. And Section (VII) concludes our work.

II. Background and State of the Art

A. Haar-like Features

Object detection utilizing Haar feature-based cascade classifiers is an effective object detection method [16].It is a machine-learning-based model where a cascade function is trained from a lot of positive and negative observations. In our experiments, images of and images without faces are the positive and negative observations, respectively.

Concretely, a general image makes up of pixel in which a person or a computer vision algorithm can recognize many distinctive shapes or patterns, partly depending on the level of contraction. During the creation of a Haar cascade, various parts of the image are cropped and scaled so that we consider only a few pixels that we call a window at a time. We will subtract some of the grayscale pixel values from others to measure the window's similarity to specific common shapes where a dark region meets a light area. If a window is very similar to one of these archetypes, it can be selected as a feature at similar positions and magnifications relative to each other. We expect to find similar features across all images of the same investigated subject.

The problem of determining the face of a person in an image is the problem of binary classification: face or not. To identify faces with Haar features, we apply Haar's features throughout the image, areas that are considered to be most similar to Haar features will be marked. Haar-like featuring is the most basic method of facial recognition. It will add a Haar feature to the whole image. The area that looks like it will identify it as the face. There will be many areas in the image where the algorithm recognizes a face. However, we do not just use a Haar feature but use a lot of features in different image positions and sizes to exclude these areas.

B. Adaptive Boosting

In machine learning [17], [18], we might take random guessing as a default baseline for evaluation reference to other learning models. A learning model is weak when it is slightly better than random guessing, while the strong one accurately classifies elements most of the time. In practice, it may not be competent to entirely rely upon the performance of just one machine learning model. Ensemble learning [19], [20] offers a systematic solution to combine the predictive power of multiple learners referring to learning a weighted combination of base models [21], [22]. The resultant model gives the aggregated output from several models. The aggregation models could be either from the same learning algorithm or different learning algorithms. The objective is to produce a classifier called C, where the output is the prediction confidence. We denote N as the number of observations. Hence, the total error E of classifier C is defined as follows:

$$E(C) = \sum_{i=1}^{N} \exp(-c_i C(\mathbf{x}_i)) \ , \tag{1}$$

where $c_i = \pm 1$ for the classification of observation $\mathbf{x}_i$. Note that we consider binary classification for a simplified discussion. In Equation (1), c_i and $C(\mathbf{v}_i)$ have the same sign if C highly confidently assign sample $\mathbf{x}_i$ with large absolute value of $C(\mathbf{x}_i)$. Consequently, its exponential error $\exp(-c_i C(\mathbf{x}_i))$ contribute very little to the total error $E(C)$ and visa versa.

AdaBoost, short for Adaptive Boosting, is widely used as an ensemble learner. It focuses on classification problems and aims to convert a set of weak classifiers into a strong one. It uses in conjunction with many other types of learning algorithms to improve performance. Conceptually, AdaBoost iteratively generates a series of classifiers C_m. In each iteration, the classifier is improved by concentrating on the misclassified observations. Therefore, each classifier is more accurate than its predecessor. It is generated as a linear combination of weak classifiers, and their coefficient gives the confidence of these weak classifiers.

Let initialize C_1 to the weak classifier $f_1(\mathbf{x})$ which misclassifies the least training samples. We denote β_1 as its coefficient, and the value is chosen concerning minimize the following error:

$$\begin{aligned} E(C_1) &= \sum_{i=1}^{N} \exp(-c_i \beta_1 f_1(\mathbf{x}_i)) \\ &= \sum_{c_i \neq f_1(\mathbf{x}_i)} \exp(\beta_1) + \sum_{c_i = f_1(\mathbf{x}_i)} \exp(-\beta_1) \ , \end{aligned} \tag{2}$$

where c_i and $f_1(\mathbf{x}_i)$ obtain the values of $+1$ or -1. Then, the difference with respect to β_1 is calculated as follows:

$$\frac{dE(C_1)}{d\beta_1} = \sum_{c_i \neq f_1(\mathbf{x}_i)} \exp(\beta_1) - \sum_{c_i = f_1(\mathbf{x}_i)} \exp(-\beta_1) \ . \tag{3}$$

We denote N_A as the number of observations that have been accurately classified. Hence, we have

$$\frac{dE}{d\beta_1} = (N - N_A)\exp(\beta_1) - N_A \exp(-\beta_1) \ . \tag{4}$$

By setting Equation (4) to zero, we solve for β_1 to find the extreme as follows:

$$\begin{aligned} \exp(2\beta_1) &= \frac{N_A}{N - N_A} \\ \beta_1 &= \frac{1}{2} \log \frac{N_A}{N - N_A} \ . \end{aligned} \tag{5}$$

As previously mentioned, the weak learners should be better than random guessing. Consequently, N_A cannot be less than $N/2$, and the more considerable value of N_A is, the more confidence it indicates. The total error through m iterations is calculated as follows:

$$\begin{aligned} E(C_m) &= \sum_{i=1}^{N} \exp(-c_i C_{m-1}(\mathbf{x}_i)) - c_i \beta_m f_m(\mathbf{x}_i) \\ &= \sum_{i=1}^{N} \exp(-c_i C_{m-1}(\mathbf{x}_i)) \exp(-c_i \beta_m f_m(\mathbf{x}_i)) \\ &= \exp(-\beta_m) \sum_{c_i = f_m(\mathbf{x}_i)} v_{i,m} \\ &+ \exp(\beta_m) \sum_{c_i \neq f_m(\mathbf{x}_i)} v_{i,m} \ , \end{aligned} \tag{6}$$

where $v_{i,m}$ is the weights.

Weight is the exponential error that the current weak classifier produces when predicting observation $\mathbf{x}_i$. Hence, Equation (6) can be split as follows:

$$E(C_m) = \exp(-\beta_m) \sum_{i=1}^{N} v_{i,m} + (\exp(\beta_m) - \exp(-\beta_m)) \sum_{c_i \neq f_m(\mathbf{x}_i)} v_{i,m} \ . \tag{7}$$

As we can see in Equation (7), the second summation depends on the classifier f_m. Consequently, the classifier which produces the sum of weights over the accurately classified observations is the largest should be selected.

Assuming that k_m is selected, we differentiate the total error concerning β_m and set the resultant difference to zero, solving for β_m. Then β_m is as follows:

$$\beta_m = \frac{1}{2} \log \frac{\sum_{c_i = f_m(\mathbf{x}_i)} v_{i,m}}{\sum_{c_i \neq f_m(\mathbf{x}_i)} v_{i,m}} \ . \tag{8}$$

The more accurate observations f_m classifies, the larger its weights are. The higher the weights are, the more confident it gets. The more assured it achieves, the larger β_m is produced.

C. Cascade of Classifiers

We will have a sequence of classifiers, in which each classifier is built using the Adaboost algorithm. Now, let all the windows go through this sequence of classifiers. The first classifier eliminates the most negative sub-windows and passes through the positive sub-window. Here, the classifier is very simple, and therefore, the computational complexity is also very low. Of course, because it is so simple, among the windows that are recognized as faces, there will be a large number of windows that are misidentified (not faces). The windows passed by the first classifier will be viewed as considered by the classifier later: if the classifier thinks it's not the face is removed, if the classifier thinks it's a face, then we pass through and move to the rear class. Later the classifier is more complex, demanding more calculation. People call sub-windows (templates) that the classifier doesn't remove as patterns that are hard to identify. The deeper these patterns get into the sequence of classifiers, the harder it is to identify. Only windows that pass through all the classifiers will decide that face.

D. Convolutional Neural Networks

The high-level general CNN architecture is illustrated in Fig. 1. Input layers are places that raw images are loaded into. The raw images are specified by their width, height, and number of channels. Typically, RGB values of each pixel form three channels. Convolutional layers apply a patch of locally connecting neurons to transform the input data from the previous layer. Consequently, the layer calculates a dot product between the area of the neurons in the input layer and the weights to which they are locally connected in the output layer. Typically, a convolution is defined as a mathematical operation describing a rule for how to merge two sets of information. Within this work, the authors briefly present the CNN background that leads to the proposed architecture in Section V-B. Further details on CNN and its next-generation investigating transfer learning [23], [24], [25], [26], [27] leaves to interesting readers.

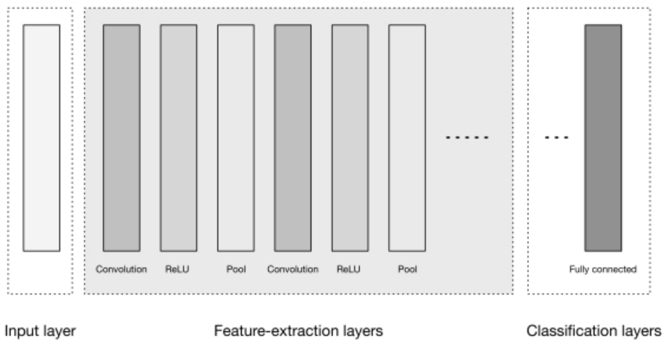

Fig. 1. High-level General CNN Architecture.

III. Hardware Components

A. Raspberry Pi 3 Model B

The Raspberry Pi (RP) is a small, cheap, single-size, single-chip computer board that comes with CPU, GPU, USB ports, and I/O pins and is capable of performing several functions. Simple as a regular calculator [28], [29]. RP 3 Model B comes with a 64-bit quad-core processor on a circuit board with WiFi and Bluetooth and USB features. It has a processing speed of 700 MHz to 1.4 GHz in which the RAM ranges from 256 to 1GB. The device's CPU is considered to be the brain of the device responsible for executing instructions based on mathematical and logical operations. The board is equipped with Broadcom video core cable, which is mainly used to play video games through the device. RP 3 comes with GPIO (General Purpose Input Output) pins that are essential for maintaining a connection with other electronic devices. These input terminals receive commands and operate based on the device program. An Ethernet port is integrated on this device to establish a communication line with other devices. The board has four USB ports that can be used to connect a keyboard, mouse, or USB 3G connection to access the internet and an SD card added to store the operating system. The power connector is a basic part of the circuit board used to supply 5 V to the board. We can use any source to set the power for the circuit board. However, we prefer to connect the power cable via the laptop's USB port to provide 5 V. HDMI port is used to connect LCD or TV, can support cables versions 1.3 and 1.4. RCA Video port is used to connect the old TV screen using 3.5mm jack without supporting the HDMI port.

B. 32 Channel USB/UART Servo Motor Controller Driver Board

A 32 Servo control circuit can be used with the software on the computer, wireless controller PS2, Android app, Arduino (or other microcontrollers) through UART communication [30]. UART stands for Universal Asynchronous Receiver/-Transmitter. It is a physical circuit in a microcontroller, or a stand-alone IC. A UART's main purpose is to transmit and receive serial data. The 32 Servo control circuit is capable of simultaneously controlling 32 servos smoothly, from which users can coordinate applications to control systems for robotic arms, industrial production lines, and machines. Servo Controller software and instruction can be found at RTROBOT[1].

Several important technical specifications of the board are summarized as follows. Physical size 63mm x 45mm. Operation voltage 5V. Servo Motor Input Voltage is 4.2V to 7.2V (According to a specific servo). CPU 32bit. Baud Rate (USB) 115200. Baud Rate (Bluetooth UART) 4800, 9600, 19200, 38400, 57600, and 115200. Flash Capacity 16M. Servo Motor Controller the Number Of Simultaneous 32. Max Action Groups 256. Control precision 1us. LEDs are dedicated to CPU power, servo motor power, and wireless remote control.

C. Camera Pi v1.3

Raspberry Pi camera has a built-in 5 Megapixel camera with high light sensitivity, able to shoot well in many different lighting conditions, both indoors and outdoors. A special feature that the camera brings is high-definition photography during movie recording. No additional USB ports are needed for the camera as the camera is securely attached to the CSI socket. This helps limit bandwidth bottlenecks for USB processing chips on Raspberry circuits. The camera cable length has been carefully calculated when it reached the required length while ensuring the speed of image transmission from the module to the RP. The important technical specifications of the Pi camera follow. Resolution 2592 x 1944 (5 megapixels). Lens f = 3.57 mm, f / 2.8. 65-degree viewing angle. Focus range 0.69m to infinity (at 1.38m). Support 1080p @ 30 fps with H.264 (AVC) codec, 720p @ 60fps and 640x480p @ 60/90 fps. CSI interface. Three dimensions are 25 mm x 25 mm x 10 mm. Weight about 2.8g.

D. Digital RC Servo LD-1501MG Motor

Digital RC Servo LD-1501MG motor is used in complex structured robots such as the humanoid robot, biped Robot, and spider robot due to the outstanding characteristics[2] compared to the traditional Analog RC Servo MG995 and MG996. RC servos or RC actuators convert electrical commands from the receiver or control system, back into physical movement. A servo plugs into a specific receiver, gyro, or FBL controller channel to move that specific part of the RC model. This movement is proportional, meaning that the servo will only move as much as the transmitter stick on your radio is moved, or as much as the gyro/FBL system instructs it to move. Digital RC Servo LD-1501MG motor has a metal gearbox, fast response speed with strong traction according to manufacturer torque figures up to 17Kg.cm. The important technical specifications of the motor are the following. The operation voltage is 5 to 7.4VDC. No-load current of 100mA, with load ¿ 500mA. Speed 0.16sec / 60° at 7VDC. Traction torque of 13Kg.cm at 6VDC; 15KG.cm at 6.5VDC; 17KG.cm at 7VDC. Weight 60g. Dimensions 40 x 20 x 40.5mm. Cable length 30cm. We use 18 motors in the implementation to achieve 4 degrees of freedom of the robot's body, which encourages us to implement complex movement and dancing.

E. Power supply and Low voltage circuit board

The AC power cord (AC) 220V is connected to L and N pins at the input on the power supply (12V, 10A). The 12V output of power adapter, V+ pin, and V- pin, are connected to positive pin (+) and negative pin (-) on the low voltage circuit board. The low voltage circuit board consists of 2 outputs; one output can be adjusted voltage from 1.25 - 32V used to power the servo motor control module, the other output is a 5V USB port used to power the RP. In this case, the appropriate voltage to supply the servo motors is 6 - 6.5V. We can adjust by rheostat on the low voltage circuit board.

F. Mechanic legs

This robot model is inspired by real spiders. However, we can only partially reproduce the movements of spiders on this robot because of the limits of joints on the robot and the complexity of the control. The simple robot model will have fewer joints, so the number of servos (actuators) will be less beneficial, which can include benefits such as reducing the weight of the robot, reducing power consumption and increasing working time The minimum condition for a robot to move is that each leg must be made up of two or more moving joints. The fact that our robot has three motion joints makes the robot move smoothly and motion control easier. The spider robot model has six legs, each of which consists of three servos (actuators) that are caught in the coal. Each side of the robot consists of three legs that allow the robot to move. The 3D sketch is done by Glovius CAD[3].

IV. Software Libraries

The Tensorflow library [31] is an open-source library developed by Google and made available to the developer community in 2015. Google Brain has developed the platform for image search, voice recognition, traces data, and many Google services. Keras [32], [33] is an open-source library for neural networks written in the Python language. Keras is a high-level API that can be used in conjunction with deep learning libraries such as Tensorflow, Theano, and CNTK. In this article, we use the Keras library in combination with the Tensorflow library [34]. OpenCV library [35] is a leading open-source library for computer vision, image processing, machine learning, and GPU acceleration features in real-time operation. OpenCV is designed for efficient computing and with a heavy focus on real-time applications. Raspberry Pi uses multiple operating systems: Raspbian, Ubuntu, Windows 10 IoT, and Chromium OS. In this article, the system uses the Raspian operating system[4] [36], [37] Raspian is the basic and most common operating system and is provided by the Raspberry Pi Foundation.

V. System Design

The system architecture can be disintegrated into three sub modules, e.g., face detection, owner recognition, and kinematic automation. The principal task in the face detection module is to detect faces from sequential pictures taken by the Raspberry camera. This task is discussed in Section V-A in more details. Then, only the cropped faces go through the process of owner

[1]https://rtrobot.org/software/
[2]https://www.rchelicopterfun.com/rc-servos.html
[3]https://www.glovius.com/
[4]https://www.raspbian.org/

recognition module where (i) the robot owners are recognized, and (ii) robot behaviors and kinematic locomotion are executed. We are going to describe these progress in Section V-B. The last module is more on the kinematics aspect, referred to Section V-C, where the authors discuss how to control the legs in case of owner detection and no owner detection. We have associated several techniques to achieve the harmonic maneuverability of detection, recognition and automation in real-time. The proposed system architecture is illustrated in Figure 2. Our hardware design and implementation of a six-legged walking robot are presented in Figure 3.

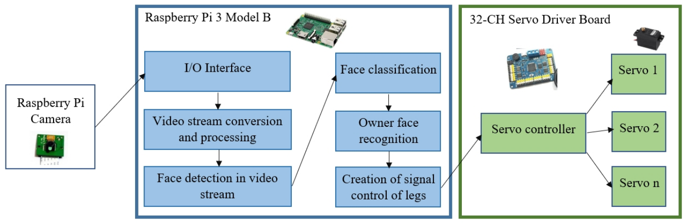

Fig. 2. The Overall Software and Hardware Architecture.

A. Module 1: Face Detection

The facial detection system is a computer application that automatically identifies a face and then identifies someone from a digital photo or a video resource. One way to do this is to compare pre-selected facial features from the image and a face database to remove non-face components. Building an automatic control program for spider robots using control signals as a result of the face recognition system with input data is frames cut from the video directly from the camera through the method of using Use the Haar Cascade technique. This RP analyzes the image and returns framed and stored faces for data training later. In general, the method is built on a combination of four components, e.g., Haar features, an integral image, Adaboost, and a cascade of classifiers. The progress of module 1 face detection is illustrated in Fig. 4.

After the faces are detected, they will be manually reclassified to verify faces, to eliminate false positives in the photo set. After the test is stored with the same number of three image files of three team members, these three files will be three data files used to train for the program in the next face recognition.

B. Module 2: Owner Recognition

In this study, the machine learning model used to classify faces is a CNN model, also known as a convolution neural network. The model has two parts: feature learning and classification. We briefly describe them in Section V-B1 and Section V-B2.

1) Feature learning: This section includes layers: Conv2D, MaxPooling2D, and Dropout. They extract the image features, learn the image features (during training), reduce the image size, and blur the image. The function of Conv2D is to extract image specifications, learn these specifications (during training). The Conv2D layers use filters to get image features (curves and lines).

The element retained during the training of extracting these properties is not the image channel. If all channels are stored, the data of the model after training will be very large. Besides, adding a new image and predicting will be no different from finding value in the database. So the values stored in this section are mostly kernels. The initial values of the kernels are randomly generated, or we can assign the initial values to them. The kernel matrix values will be updated during the Backpropagation process.

2) Classification: This part is two fully connected layers, which consists of 1 dense layer containing 512 nodes and one dense layer consisting of 3 nodes, calculating values and make a classification. Note that the output contains three nodes because we set the number of robot's owners is three. The values stored after training in this part are the weights that are updated during the backpropagation process. The sigmoid activation function is used in the last layer to predict the percentage of an object with an input sample. The class with the highest score will be the final prediction.

We proposed the CNN architecture presented in Fig. 5. We implemented the experimental model in Keras and TensorFlow. The model consists of four convolution Conv2D layers. The first, second, and fourth convolution layers accompany a MaxPooling2D a Dropout layers. The third convolution layer is just attached to a MaxPooling2D layer. The last two layers are dense. Default Adam [38] is the optimization method. Categorical cross entropy [39], [40] is the loss measurement.

C. Module 3: Maneuverable Automation

A multi-legged robot [41], [42] maintains a tremendous potential for maneuverability over different terrain, particularly in comparison to wheeled robot. It possesses less complexity than a human-like walking robot regarding stability and development effort. A six-legged robot inspired by biology is one of the most common design [43], [44]. It is important to develop a good control mimicking the kinematic behavior of the complex six-legged robotic mechanism. The overall design of our proposed automation is presented in Fig. 7. We use the OpenCV library to process data received from the RP camera. The system selects one frame out of five received frames. The reason is that the robot takes a certain amount of time to complete the current motion, then we perform the identification and send the next control signal. During the experiment, we found that this ratio gives the most natural motion of the robot. When there is a frame, the system will detect the face cut, if any. These cropped faces are then processed by the machine learning model developed on the Keras and TensorFlow library. Next, the system controls the movements of the robot according to the predicted result. The system checks the current rotation angle of each servo motor and finds the appropriate pulse value for the action to be performed. The control signal will be transmitted to each motor, and the robot forms a motion.

We program the robot to perform some basic movements based on the results of the face recognition of the owner. The specific movements of the robot are as follows. If the owner is Hoai Bao, the robot performs the stand-up move. If the owner is Hung Tam, the robot performs the sit-down operation. If the owner is Nam Tran, the robot will move towards the owner. If

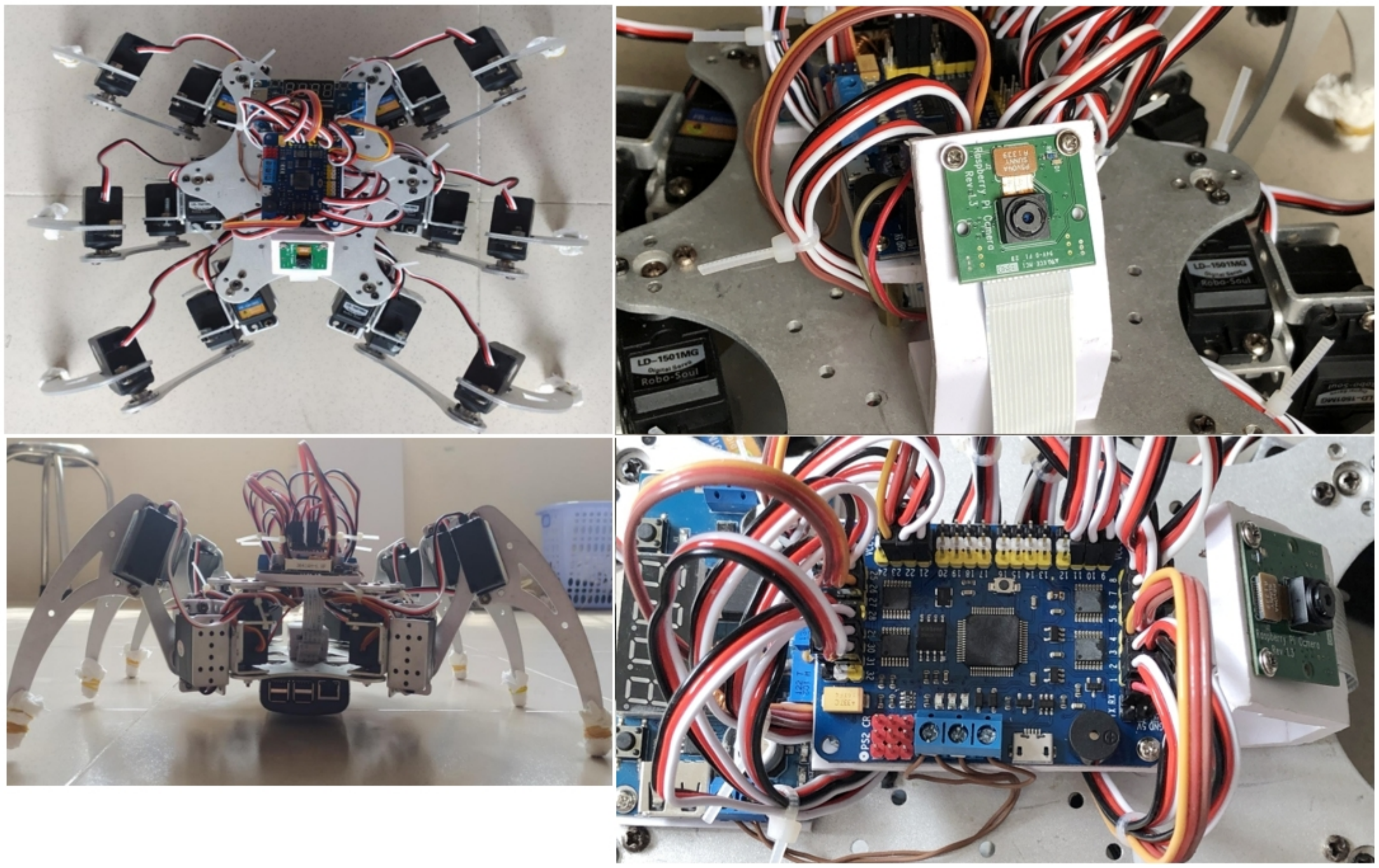

Fig. 3. Several Views of the Six-legged Walking Robot.

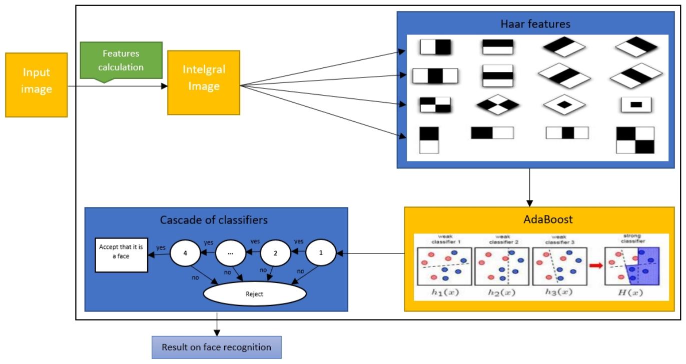

Fig. 4. The Architecture of Module 1: Face Detection.

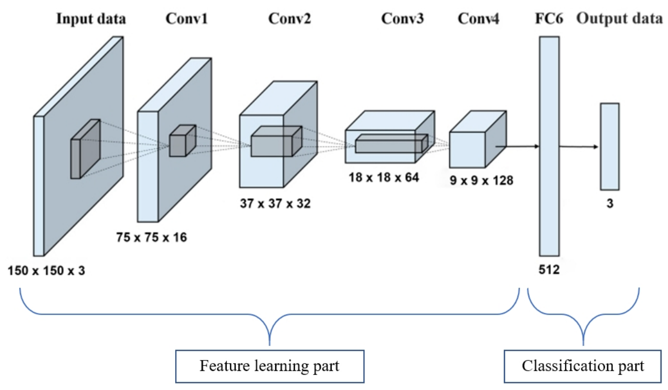

Fig. 5. The Architecture of Module 2: Face Recognition.

one of the three owners is not recognized, keep the previous movement. To control the rotation of the servo motor, we make a change in the pulse value. For LD-1510MG servo motors used in this study, the pulse range is from 500 to 2400 us. Fig. 6 depicts the pulse and sweep angle of the motor.

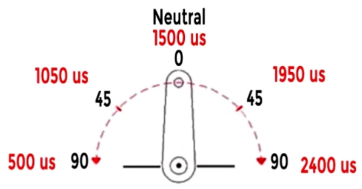

Fig. 6. Motion Angle of a Motor.

VI. Experiments

A. Hardware and Software Setup

Our six-legged walking robot consists of several hardware components such as one RP 3 model B, one 32 channel USB/UART servo motor controller driver board, one RP camera

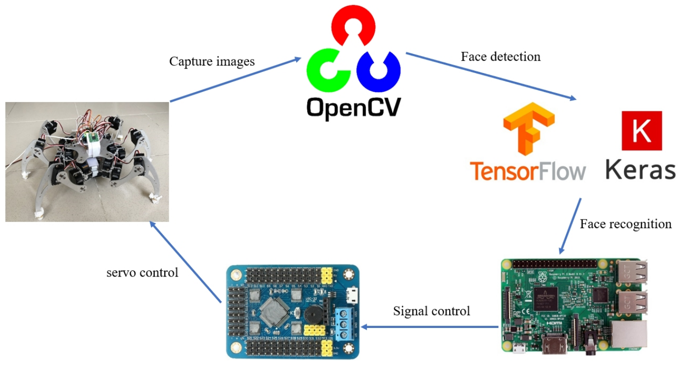

Fig. 7. The Architecture of Module 3: Maneuverable Automation.

v1.3, 18 digital RC servo LD-1501MG motors, one power supply 12V 10A, one low voltage circuit board, mechanic legs, and other components. The hardware connection is presented in Table I. The RP 3 model B is considered the robot brain where most of the controls are calculated from here.

TABLE I. CONNECTIONS OF MAIN HARDWARE DEVICES

Device	Connector Type	Device
Raspberry Pi	CSI - CSI	Pi Camera v.1.3
	RX - TX	32 Channel USB/UART Servo Motor Controller Driver Board
	TX - RX	
	GND - GND	
32 Channel USB/UART Servo Motor Controller Driver Board	GND - GND	Servos
	VCC - VCC	
	heart beat signals	

B. Legs Controller

Each servo motor is then connected to the control circuit. For ease of programming and maintenance, we specify the wiring order as follows: 6 front two-servo motors will be connected to pin 1 to 6 of the motor control module. The six servo motors of the middle two legs will be connected to pin 7 to 12. And the last six servo motors of the rear legs are connected to pin 13 to 18. To check whether the servo motors are correctly installed, we connect the servo motor control module to the computer via the USB interface and use the software[5] to check the operation of each servo.

In the program, after having information on face recognition, this signal is the input for the control signal for the spider robot. The signals are transmitted except the Raspberry Pi to the servo control circuit by UART communication. Now we need to determine the pulse level for each leg of the robot is the servo. The balance joints smoothly work together, similar to the movements of spiders, in reality, to shape the movements of the spider robots. The control signals are transmitted in the form of pulses with an execution time of ms. Example #1P2500T100 where 1 is the pin position 1, P2500 means 2500 pulses, and T100 is the execution time of 100ms. The simulation of robot moving forward, backward and turning-left is presented in Fig. 8, 9, and 10.

Control angles are calculated manually using RTROBOT control simulation software. We check and select the appropriate pulse width to create motion for 18 robot leg joints. The settings are saved into a configuration file containing foot control information and pulse value for each motor. Activities are grouped into blocks. The time taken for a moving group is fixed the same by 1 second to avoid control signal congestion. After detecting a face, the RP calls control signals that are pre-configured according to each face we have previously classified. The control signal is executed in series, but the execution speed of the foot joints is fast so that it will see simultaneous movements.

[5]https://rtrobot.org/software/

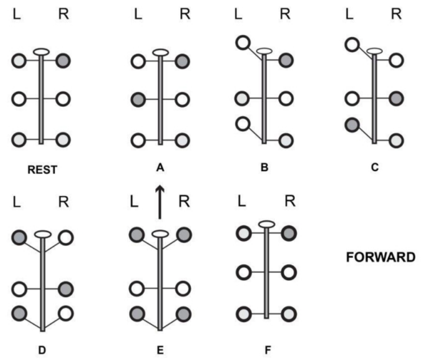

Fig. 8. Simulation of Robot Moving Forward.

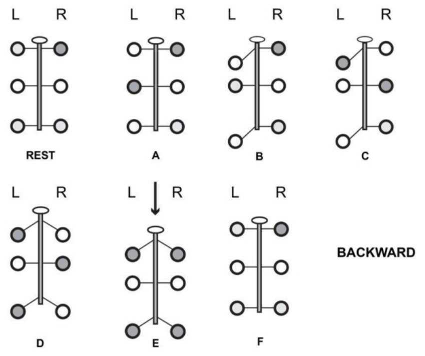

Fig. 9. Simulation of Robot Moving Backward.

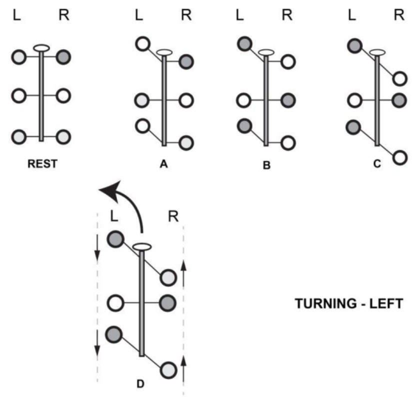

Fig. 10. Simulation of Robot Turning Left.

C. Face Detection and Image Collection

To apply the Haar recognition method with Adaboost, we use the *detectMultiScale* function to search for faces in the image. Each image will be scanned in turn, and the program will confirm whether it is a human face or not and determine the number of faces in that frame by sliding the Haar symbols on the image, helping us identify All faces in the photo. The *detectMultiScale* function is a face search, which is part of the *CascadeClassifier* class. The codes are presented in Listing (1).

```
cascade = cv.CascadeClassifier(
haarcascade_frontalface_default.xml)
gray = convert_to_gray(imagePath)
faces = cascade.deetectMultiScale( gray,
scaleFactor = 1.1,
scaleNeighbor =5, minSize(5,5), maxSize
(100,100))
```

Listing 1: Detect faces with detectMultiScale function.

The .xml file in Listing (1) contains the classifiers built on the standard database sets that is part of the OpenCV library.

The face detection on video is similar to the image because the video is a series of frames. Then we will capture each frame and save it. The program will read the captured frames again. Then we let the face detection program run on all image data and save all identifiable faces into another folder with the purpose of building data for the training set to be used in the recognition face. The codes in Listing (2) present how images are captured from Pi camera.

```
cap=cv.VideoCapture(0)
i=0
while (i!=15):
frame=cap.read()
frame=cv.flip(frame,-1)
cv.imshow("Frame",frame)
name= "\%s" %(i)
name= name + ".png"
i=i+1
cv.imwrite(name,frame)
cap.release()
cv.destroyAllWindows()
```

Listing 2: Images are captured using Pi camera.

Faces after being detected, the image will be manually reclassified to check again to identify faces correctly, removing false identities included in the image file. After the test is stored with the same number of three image files of three team members, these three files will be three data files used to train for the program in the next face recognition. Through experiments with model *haarcascade_frontalface_default.xml* with images containing human faces, the program recognized at a good level.

To increase the data for training, we use the solution augmentation technique to provide more data. *ImageDataGenerator* function is used in the implementation. The codes is presented in Listing (3).

```
image_gen = ImageDataGenerator(rescale
=1./255, horizontal_flip = True,
zom_range=0.25, rotation_range=45)
train_data_gen = image_gen.
flow_from_directory(batch_size=1000,
directory="C:\\Test\\Image\\output\\",
shuffle=True,
target_size=(IMG_HEIGHT, IMG_WIDTH))
```

Listing 3: ImageDataGenerator function for augmentation.

D. Face Recognition

1) Datasets: The experimental dataset is a set of images of the face of the robot's owners. This volume is divided into three parts, one part is used to train the machine learning program (training set), the other part is used for validation (validation set), and one part is used to check the accuracy. The original image's resolution taken by Pi Camera is 2048 x 1536. Then face regions are detected and cropped to 150 x 150.

The training set consists of 2250 photos divided into three folders. Each folder contains pictures of a specific person (Hoai Bao, Nam Tran, Hung Tam). These samples were taken from Pi cameras in environments with different light intensity and different angles. Besides, we also use augmentation methods such as: rotating the image with an angle of x degrees (with x being the number of degrees), enlarging the image, zooming the image, flipping the image, or combining many of them to get more photo patterns and to avoid overfitting. The validation set of 750 photos is also divided into three folders. Each folder contains pictures of a specific person (Hoai Bao, Nam Tran, Hung Tam). These images are also taken from the Pi camera. However, they are received at random. These samples will be reviewed after each epoch (number of model training) as one of the model's goodness test metrics. Similar to the validation set, the test set of 741 images is also divided into three folders corresponding to three people who are considered the owners of the robot.

2) Evaluation Metrics: Suppose we are solving a binary classification task with a labeled dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}$. Given a threshold parameter ϕ that guilds our decision rule $g(\mathbf{x})$ and count the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN). We also define m_+ the total of condition positives, m_- the total of condition negatives, $\hat{m}_+$ the total predicted condition positives, $\hat{m}_-$ the total predicted condition negatives, and m the total population.

We can compute the sensitivity, also known as true positive rate (TPR) or recall by using:

$$\text{TPR} = \frac{\text{TP}}{m_+} \approx P(\hat{y} = 1 | y = 1) \ . \tag{9}$$

Similarly, we can compute the fall-out, also known as false positive rate or probability of false alarm by using:

$$\text{FPR} = \frac{\text{FP}}{m_-} \approx P(\hat{y} = 1 | y = 0) \ . \tag{10}$$

The true negative rate (TNR) or specificity is defined as follows:

$$\text{TNR} = \frac{\text{TN}}{m_-} \approx P(\hat{y} = 0 | y = 0) \ . \tag{11}$$

The false negative rate (FNR) or miss rate is calculated as follows:

$$\text{FNR} = \frac{\text{FN}}{m_+} \approx P(\hat{y} = 0 | y = 1) \ . \tag{12}$$

If we work with a dataset for binary classification when the number of negatives is very large or a dataset for multi-class text prediction when a class imbalance exists, considering TPR, FPR, TNR, and FNR themselves is not very informative. Before going further, we define positive predictive value (PPV) or precision as follows:

$$\text{PPV} = \frac{\text{TP}}{\hat{m}_+} \approx P(y = 1 | \hat{y} = 1) \ . \tag{13}$$

By combining Equation (9 and 13), we can compute F1-score as follows:

$$\text{F1-score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \ . \tag{14}$$

3) Evaluation Results: Model trained on a 2-core 4-thread laptop, 12GB RAM, Nvidia GTX 950M GPU. The size of each batch is 32 images. Model converges after 10 epochs.The model is then tested on test data to evaluate its final accuracy. The TP, FN, and FP over the test set are presented in Table II.

TABLE II. THE TP, FN, AND FP OVER THE TEST SET

No.	Owner face	TP	FN	FP
1	Nam Tran	247	16	8
2	Hung Tam	247	12	5
3	Hoai Bao	247	15	3
Total		741	43	16

From the evaluation result presented in Table II, we can easily calculate Precision ≈ **0.98**, Recall ≈ **0.95**, and F1-score ≈ **0.96**.

4) Experimental Remarks: Model training can be done on Raspberry or regular computers. Because RP is an embedded computer, the hardware configuration and processing speed are relatively limited. Training on Raspberry takes a lot of time. In this case, the training time for one epoch is more than 2 hours (the model used in this topic is ten epoch training) for a training set of 2250 images, and each image size is 150x150. Besides, we have to turn off all the applications so that the training program has enough Ram to execute. However, we still achieved experimental results similar to those done on laptops. During experiments, we also try the VGG-16 model [45], which is very lightweight to compare the prediction accuracy with our proposed CNN model. However, the VGG-16 model cannot fit into RP's memory when loading images. Note that the image's resolution is only 150 x 150.

VII. CONCLUSIONS

Throughout the article, the authors present Haar-like features, ensemble learning, and CNN as the methodology background. We discuss hardware components and mechanical legs, which build a six-legged working robot. The critical contribution is the proposed system design, where three sub modules, e.g., face detection, face recognition, and kinematic automation, work harmonically in real-time. Therefore, a complete solution associating both mechanical moving parts of a walking robot, IoT hardware components, and software has been adequately demonstrated. The proposed design and implementation has orchestrated several complex tasks. First, we address the task of interlocutor face detection and recognition, utilizing ensemble learning and convolutional neural networks. Second, we develop maneuverable automation of a six-legged robot via hexapod locomotion. And last but not least, we deploy the proposed system on a Raspberry Pi that has not been previously reported in the literature. The experiments are well described to encourage further reproducibility and improvement.

REFERENCES

[1] E. Celaya and J. M. Porta, "A control structure for the locomotion of a legged robot on difficult terrain," *IEEE Robotics & Automation Magazine*, vol. 5, no. 2, pp. 43–51, 1998.

[2] B. Klaassen, R. Linnemann, D. Spenneberg, and F. Kirchner, "Biomimetic walking robot scorpion: Control and modeling," *Robotics and autonomous systems*, vol. 41, no. 2-3, pp. 69–76, 2002.

[3] C. Loughlin, S. Soyguder, and H. Alli, "Design and prototype of a six-legged walking insect robot," *Industrial Robot: An International Journal*, 2007.

[4] J. Auerbach and J. C. Bongard, "How robot morphology and training order affect the learning of multiple behaviors," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 39–46.

[5] S. S. Roy and D. K. Pratihar, "Dynamic modeling, stability and energy consumption analysis of a realistic six-legged walking robot," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 2, pp. 400–416, 2013.

[6] ——, "Kinematics, dynamics and power consumption analyses for turning motion of a six-legged robot," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, pp. 663–688, 2014.

[7] G. Zhong, H. Deng, G. Xin, and H. Wang, "Dynamic hybrid control of a hexapod walking robot: Experimental verification," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 8, pp. 5001–5011, 2016.

[8] T. Maiti, Y. Ochi, D. Navarro, M. Miura-Mattausch, and H. Mattausch, "Walking robot movement on non-smooth surface controlled by pressure sensor," *Adv. Mater. Lett*, vol. 9, no. 2, pp. 123–127, 2018.

[9] F. Tedeschi and G. Carbone, "Design of a novel leg-wheel hexapod walking robot," *Robotics*, vol. 6, no. 4, p. 40, 2017.

[10] V. Patchava, H. B. Kandala, and P. R. Babu, "A smart home automation technique with raspberry pi using iot," in *2015 International conference on smart sensors and systems (IC-SSS)*. IEEE, 2015, pp. 1–4.

[11] A. Pajankar, *Raspberry Pi computer vision programming*. Packt Publishing Ltd, 2015.

[12] I. Iszaidy, R. Ngadiran, R. Ahmad, M. Jais, and D. Shuhaizar, "Implementation of raspberry pi for vehicle tracking and travel time information system: A survey," in *2016 International Conference on Robotics, Automation and Sciences (ICORAS)*. IEEE, 2016, pp. 1–4.

[13] C. Kaymak and U. Aysegul, "Implementation of object detection and recognition algorithms on a robotic arm platform using raspberry pi," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE, 2018, pp. 1–8.

[14] J. Cicolani, *Beginning Robotics with Raspberry Pi and Arduino: Using Python and OpenCV*. Apress, 2018.

[15] K. Janard and W. Marurngsith, "Accelerating real-time face detection on a raspberry pi telepresence robot," in *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*. IEEE, 2015, pp. 136–141.

[16] A. Sharifara, M. S. M. Rahim, and Y. Anisi, "A general review of human face detection including a study of neural networks and haar feature-based cascade classifier in face detection," in *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*. IEEE, 2014, pp. 73–78.

[17] A. C. Faul, *A Concise Introduction to Machine Learning*. CRC Press, 2019.

[18] N. Duong-Trung, *Social Media Learning: Novel Text Analytics for Geolocation and Topic Modeling*. Cuvillier Verlag, 2017.

[19] L. Nanni, S. Brahnam, and A. Lumini, "Face detection ensemble with methods using depth information to filter false positives," *Sensors*, vol. 19, no. 23, p. 5242, 2019.

[20] Z. Wu, W. Lin, and Y. Ji, "An integrated ensemble learning model for imbalanced fault diagnostics and prognostics," *IEEE Access*, vol. 6, pp. 8394–8402, 2018.

[21] A. Saeed, A. Al-Hamadi, and A. Ghoneim, "Head pose estimation on top of haar-like face detection: A study using the kinect sensor," *Sensors*, vol. 15, no. 9, pp. 20 945–20 966, 2015.

[22] S. W. Cho, N. R. Baek, M. C. Kim, J. H. Koo, J. H. Kim, and K. R. Park, "Face detection in nighttime images using visible-light camera sensors with two-step faster region-based convolutional neural network," *Sensors*, vol. 18, no. 9, p. 2995, 2018.

[23] N. Duong-Trung, L.-D. Quach, and C.-N. Nguyen, "Towards Classification of Shrimp Diseases Using Transferred Convolutional Neural Networks," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 4, pp. 724–732, 2020.

[24] ——, "Learning deep transferability for several agricultural classification problems," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 1, 2019.

[25] N. Duong-Trung, L.-D. Quach, M.-H. Nguyen, and C.-N. Nguyen, "A combination of transfer learning and deep learning for medicinal plant classification," in *Proceedings of the 2019 4th International Conference on Intelligent Information Technology*, 2019, pp. 83–90.

[26] R. Zhang, H. Tao, L. Wu, and Y. Guan, "Transfer learning with neural networks for bearing fault diagnosis in changing working conditions," *IEEE Access*, vol. 5, pp. 14 347–14 357, 2017.

[27] N. Duong-Trung, L.-D. Quach, M.-H. Nguyen, and C.-N. Nguyen, "Classification of grain discoloration via transfer learning and convolutional neural networks," in *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing*, 2019, pp. 27–32.

[28] S. Monk, *Raspberry Pi cookbook: Software and hardware problems and solutions*. " O'Reilly Media, Inc.", 2016.

[29] D. Molloy, *Exploring raspberry PI*. Wiley Online Library, 2016.

[30] "Basics of uart communication," Apr 2017. [Online]. Available: https://www.circuitbasics.com/basics-uart-communication/

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[32] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[33] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[34] J. Brownlee, *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery, 2016.

[35] A. F. Villán, *Mastering OpenCV 4 with Python: a practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4 and Python 3.7*. Packt Publishing Ltd, 2019.

[36] W. Harrington, *Learning raspbian*. Packt Publishing Ltd, 2015.

[37] P. Singh, P. Nigam, P. Dewan, and A. Singh, "Design and implementation of a raspberry pi surveillance robot with pan tilt raspbian camera," *International Journal of Nanotechnology and Applications*, vol. 11, no. 1, pp. 69–73, 2017.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[39] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Advances in neural information processing systems*, 2018, pp. 8778–8788.

[40] R. Gómez, "Understanding categorical cross-entropy loss, binary cross-entropy loss, softmax loss, logistic loss, focal loss and all those confusing names," *URL: https://gombru.github.io/2018/05/23/cross_entropy_loss/*, 2018.

[41] S. Peng, X. Ding, F. Yang, and K. Xu, "Motion planning and implementation for the self-recovery of an overturned multi-legged robot," *Robotica*, vol. 35, no. 5, pp. 1107–1120, 2017.

[42] A. Mahapatra, S. S. Roy, and D. K. Pratihar, "Multi-legged robots—a review," in *Multi-body Dynamic Modeling of Multi-legged Robots*. Springer, 2020, pp. 11–32.

[43] B. He, S. Xu, Y. Zhou, and Z. Wang, "Mobility properties analyses of a wall climbing hexapod robot," *Journal of Mechanical Science and Technology*, vol. 32, no. 3, pp. 1333–1344, 2018.

[44] G. Zhong, L. Chen, Z. Jiao, J. Li, and H. Deng, "Locomotion control and gait planning of a novel hexapod robot using biomimetic neurons," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 624–636, 2017.

[45] Y. T. Li and J. I. Guo, "A vgg-16 based faster rcnn model for pcb error inspection in industrial aoi applications," in *2018 IEEE international conference on consumer electronics-Taiwan (ICCE-TW)*. IEEE, 2018, pp. 1–2.