Doctoral Dissertation

Academic Year 2019

Self-recovery Strategy for Multi-legged Robot with Damaged Legs

Division of Human Mechanical Systems and Design,
Graduate School of Engineering,
Hokkaido University

Sarun Chattunyakit

HOKKAIDO
UNIVERSITY

A Doctoral Dissertation
submitted to Graduate School of Engineering, Hokkaido University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Engineering

Sarun Chattunyakit

Thesis Committee:

Professor Yukinori Kobayashi	(Supervisor)
Professor Itsuro Kajiwara	(Member)
Professor Atsushi Konno	(Member)

Abstract of Doctoral Dissertation of Academic Year 2019

Self-recovery Strategy for Multi-legged Robot with Damaged Legs

Summary

Due to the flexibility of multi-legged robots, they can be applied in several applications in which regular wheel-based robots are not suitable to perform. However, the multi-legged robots require a number of sensors and actuators. In a normal case, legged robots can perform properly with a controller programmed by the user, but there are some failures occurring when some sensors and actuators are no longer available. The legged robots are nonfunctional after getting damaged since the prior control strategies cannot be employed to operate efficiently with transferred models. A self-recovery method can overcome this problem by finding alternative behavior of the robot. Although these techniques make robots possible to walk after damaging, the existing methods require a lot of time to operate. This research mainly focuses on developing the algorithm for self-recovery, concerning about the learning time and complexities. Not only the self-recovery method is studied in this research, but also other factors that can help the robot to be able to move again.

The novel structure model of the quadruped robot has been proposed in this study. The caterpillar-inspired quadruped robot (CIQR) is developed to imitate the caterpillar crawling locomotion when the robot has a small number of active legs. The caterpillars' proleg is added on the robot limb to improve the ability to move after some parts of the robot got damaged. This lets the legged robot become movable even if it has only one leg. However, the structure of legs has to be designed circumspectly due to the face that the proleg can limit the reachable space of robots leg while operating with normal quadruped gaits. In this paper, the new shape of the robotic leg is designed with the inspiration of caterpillar and optimized using PSO algorithm. The fitness function of PSO is set as the distance that the robot can travel in both crawling and trotting gait.

To discover the new model, the PSO-based Leg-loss Identification method (PLI) is proposed. The PLI method uses only onboard sensors that let the robot become more versatile. Particle swarm optimization is utilized to optimize the

fitness function that is set as the resemblance of candidate models and actually damaged the robot. The acoustic-based fault diagnosis for legged robots (AFL) is developed to detect the abnormalities of joints. The sound of servo motors is recorded simultaneously while a multi-legged robot is executed to perform specific actions. The results show that both proposed methods can detect the fault parts properly with the broken robot in the experiments.

Moreover, the development of new bio-inspired locomotion method is conducted to help the legged robot that has a small number of legs to be able to move again after getting damage. The concept is based on the movement of mudskipper in nature. Self-learning mudskipper-inspired crawling method (SLMIC) is proposed in this study. The reinforcement learning method, Q-learning, is integrated to improve the adaptability of locomotion. The results show that the proposed method is feasible to employ a damaged robot that has two legs.

According to the results of proposed works, the upshot of this study is the possibility that the methodology used in this study is feasible to implement with the damaged legged robot in a practical situation.

Keywords:

Legged Robot, Damage Detection, Reinforcement Learning, Gait Adaptation, Fault Tolerance

Graduate School of Engineering, Hokkaido University

Sarun Chattunyakit

Acknowledgements

Entering a Ph.D. course is like the toughest thing in my life. It seems impossible for me to complete this study without many supports from all those who help me in any respect. Firstly, I would like to express my special appreciation and thanks to my supervisor Prof. Yukinori Kobayashi, who accepted me as his advisee. What he did is not about the study and research, but he also gave me a lot of support during my stay in Japan. I have learned a lot from him. Without his guidance and feedback, I am quite sure that I cannot achieve the Ph.D. as I expected. Secondly, I greatly appreciate the support received from Assoc. Prof. Takanori Emaru who gave me many suggestions and feedbacks for my research. He also helped me in supporting my daily life in Japan. Thirdly, I would like to show my appreciation to Asst. Prof. Ankit Ravankar, who help me check my paper and give me suggestions.

Moreover, I would like to thank the Robotics and Dynamics Laboratory members and alumni, who always support and cheer me up when I got depressed, especially, Dr. Koichi Imaoka, Dr. Jixin Lv, Masaru Sakuma, Haruka Tanabe, Yoshiki Aoyama, and Takuma Okawara. Many thanks also to my Thai friends, Asst. Prof. Pawit Tangviroon, Dr.Shanya Orasutthikul, Dr.Apichaya Aneksampant, Namo Podee, Teeranai Srimahachota, and Sirawich Vachmanus, who supported me during my stay in Japan. I am grateful to staffs of the Graduate School of Engineering for their kind assistance and warm support, especially, when I am living overseas.

Finally yet importantly, I would like to thank my family and my parents who always support me and stay here by my side.

This work is supported by MEXT (Ministry of Education, Culture, Sports, Science, and Technology), Japan.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Research Background

Legged robots can be applied in several applications (e.g. being in action on uneven terrain and outdoor environment) on account of flexibility in which regular wheel-based robots are not attainable. However, intricate control manners and sophisticated components are necessary to achieve the desired behaviors. In general, legged robots can function successfully with predesigned controllers. However, there are some failures occurred when some parts of robots are not working, such as broken legs and joints lock. Damaged legs can lead to a lack of success while operating with the prior-designed controller because of changed models. Recently, self-damage recovery algorithms have been proposed so far to overcome this drawback. It makes a robot more flexible as allowed the robots to create new-alternative behaviors to deal with broken robots automatically without human control. Human and certain animals, in like manner, will be able to learn the new gait when their legs get injured. A biped dog can create a distinctive self-recovery behavior with only two hind legs as it can do balancing, standing and even walking. Thus, it can be recapitulated that applying self-damage recovery behavior can open doors of possibility for damaged-legged robots to restore themselves.

As self-recovery can provide robustness and efficiency, many types of research and approaches are proposed by means of developing the learnable systems. The reward function is an important component of the learning system. Several researchers use fruits, snack or juice as the reward to motivate animals when they have learned new things. Policy gradient methods employ the stochastic optimization method to find a local highest value of the reward function [78]. In the beginning, several random controllers are generated, designed by user or demonstration results. The gradients of reward function are estimated in the control space. After that, the parameter values of the controller are modified by gradient

information. The result will be accepted when finding a satisfying controller or becoming converges. Many kinds of research have successfully applied the gradient method with robotic systems. Kimura et al. use this method with a quadruped robot with 8-DOFs and 72 control parameters [47]. It spends 80 minutes for learning to walk. Another quadruped robot developed by Kohl and Stone is also applied by set starting behavior as walking. After 3 hours, the robot has already learned with 12-DOFs and 12 control parameters [48]. Consisting of 2-DOFs and 46 control parameters, biped-robot spends only 20 minutes to learn, and it starts with standing behavior [82]. Following the existing experiments, it seems that learning time is caused by the number of parameters. In addition, the gradients are estimated in different ways for each approach. Evolutionary algorithms (EAs) are another technique that is used to solve for the optimal function of stochastic optimization methods [18]. Most of the existing methods of EAs begin with creating populations of a candidate solution, then they are initialized randomly. The performance of each controller is evaluated and sorted, after that, the most efficient controllers will be selected to create a new population for the next iteration. Even though EAs can be applied successfully with robots, the main obstacles of this approach are the learning time and the number of evaluation. Quadruped robot consisting of 9-DOFs and 5 control parameters spends 2 hours for learning time [103], and Hexapod robot with 12-DOFs takes 10 hours for learning time [7]. In addition, most approaches employ the external reward for learning system, such as an electrical signal. However, using the internal reward is better in case that robots are nonfunctional in a practical situation. Hornby et al. have developed the algorithm for the quadruped robot with internal reward, but learning time is so long up to 25 hours.

The insect-inspired central pattern generators are employed in a robot with leg malfunction successfully [65]. The controlling frequency of each leg is changed to compensate for the leg malfunction. In the experiment, the legs of the robot are disabled in order to mimic the leg malfunction. However, the broken legs, joint, and body have not been considered yet in [65]. In [49], the six-legged robot will be able to walk after getting damaged; still, it cannot cope with some imperfections owing to the non-updated model. Bongard and et al. published the self-modeling method that lets the robot be able to reconstruct an exact model [7]. The robot can discover the current model using the collation of responses between the actual robot and candidate models in the simulation. The candidate models in the search space will evolve their body in which makes their behaviors

similar to the real-damaged robot. The optimization method, genetic algorithm (GA), is utilized for this process. The responses of the damaged robot are observed using the external sensory device, the Wii infrared remote. After applying this approach, the new proper model and alternative behaviors have been found. The result shows that, with self-modeling, the robot can successfully learn a new walking gait after losing one leg. Self-model is identified by means of active learning loop. Action selection, aiming to select the best action that can differentiate the model, is the first state, and the 36 possible actions with the population of 16 candidate models are experimented to observe the orientation of robot's body. The exact orientation of the robot is measured by an external camera. The selection of the action is done by choosing the most disagreed model. If the action selected is less 15 actions, the action selection loop will be performed again. On the other hands, if the action is found properly, an optimization algorithm will be used to find the optimal solution of the population of the model, and then the best model is found. For control optimization, an EA is used to optimize a population of candidate controller, determined by the forward displacement. The best controller will be found in simulation and transferred to the robot. In addition, these experiment of the starfish-like robot does not rely on internal sensors which are important when the robot is damaged in unreachable areas. However, using an external sensor limits the ability of the legged robot to perform in other different environments. To eliminate this problem, the algorithm proposed in [51] employs only built-in sensors, i.e., tilt sensor and switches, to observe its behavior, called a self-identification method based on biological evolutionary mechanisms. The damaged robot can obtain the new model successfully with this method. In addition, this method is applied to get alternative gaits in [26]. However, GA can provide resolution errors as encoding and decoding processes are required as dealing with binary chromosomes. Moreover, due to a limited number of sensors, some candidate models with different leg's length can yield the duplicate final postures at the end of the process so that the candidate models in search space have to perform twice to overcome this problem.

Fault diagnosis and model identification are therefore crucial to prevent legged robots from the unanticipated event caused by faults. Joint fault problems are one of the accidental issues occurring in legged walking robots. The robots will not be able to walk properly when the joints of the robots' legs become uncontrollable. In [66], the robot uses current sensors and Inertial Measurement Unit (IMU) to identify the failures of itself. Although the problem statuses indicate

correctly in the experiments, a number of current sensors are required to measure the amounts of current spent. Image processing, recognizing artificial markers attached to robots' legs, is developed to detect the defective actions as well [70]. The robot is programmed to execute its legs into three different levels to observe irregular behaviors. In the event of sightless markers caused by accidents, i.e., concealment of surrounding objects or detachment of artificial markers, this method will diagnosis the fault ineffectively. The Least-Mean-Squares (LMS) filter-based approach is developed so far as to discover the reliability of the joints [56]. This manner can identify the locked joints sufficiently; however, the non-linearity of the system is not considered and the outputs of each motor are required to measure for differentiation procedure. Due to the limitations of existing methods as above, acoustic processing can be put into action to perform the joint fault diagnosis.

1.2 Motivations and Contributions

In case of damage, the performance of legged-robot will be limited by the effect of unusable components. The robot is no longer controlled by the same aspect as an undamaged robot. Fixing the damaged robot is barely possible when it performs in unreachable areas. Thus, it will be better if the robot can learn by itself to find other alternative solutions. Moreover, the nature-inspired concept is considered to combine with machinery system to increase the flexibility. Recent techniques make it possible for a robot to discover a new behavior after damage, but most of them require a large number of iterations to train the robots. Self-modeling lets robot be more flexible by creating its own model. However, there are some problems occurred when testing with a real damaged robot. If a long learning time and system failures can be minimized, the risk of damage will be reduced. As performing time will be limited by the capability of battery equipped on the robot, fast learning is more suitable in this case. Most existing methods use external rewards as the main contribution to the learning system, causing the robots to be unusable to train themselves in practical situations. It can be noticed that this system is not limited especially for the legged robot, but it can be employed with other learnable systems.

The contribution of this study is detailed as follows:

- The novel structure model of the quadruped robot has been proposed. The caterpillar-inspired proleg is added on the robot limb to improve the ability

to move after some parts of the robot got damaged. This lets the legged robot become movable even if it has only one leg.

- Self-modeling method with internal sensor feedback is developed to let robot become function-able in an unfamiliar environment. And the acoustic-based joint fault detection method has been investigated. The robot can perform specific actions, listen to its sound, and then diagnosis the fault of the robot's joints.
- The development of new bio-inspired locomotion method is conducted to help the legged robot that has a small number of legs to be able to move again after getting damage. The concept is based on the movement of mudskipper in nature. Moreover, the reinforcement learning method is integrated with the proposed method to make it adaptable.

1.3 Limitations and Delimitation

Since the problem is one of the most challenging topics in the research fields of robotics, it becomes impossible to cover all of the aspects related to this research. Hence, it is denoted that this study mainly focuses on the self-recovery method for the legged robot with a small number of active legs (less than 4 legs). Only the servo motor-based legged robots are considered in this study. Accordingly, two different types of faults are studied in this work which are leg loss and joint lock problems. Last but not least, the sensory problem is not considered in this study.

1.4 Dissertation Outlines

The first chapter gives a brief overview of the research background. Motivation and contribution are also written here. After that, the limitation and delimitation of this study are detailed. The development of caterpillar-inspired structure is described in the second chapter. The third chapter analyzes fault detection methods for the legged robot with broken legs and joint motors. In the fourth chapter, a new methodology for self-recovery based on the inspiration of mudskipper locomotion is presented. Some conclusion and future work are drawn in the final chapter.

Chapter 2
Design of Quadruped Robot

2.1 Introduction

In recent years, legged robots have been widely utilized in several applications due to the fact that legged robots are more flexible than wheel-based robots in terms of mobility and energy efficiency [30]. Types of the legged robot can be categorized by many methods, such as the number of legs, locomotion, and application. In [83], example types of the legged robot were reported based on the number of legs, as monopod robots, biped robots, and multi-legged robot. Monopod robots are one-leg robots with the ability to hop for locomotion. They have low dimensionality and mostly without static stability [23]. In the past, there were several kinds of monopod robots and their control methods that have been developed and researched. ARL Monopod II was designed with a prismatic leg at the hip joint and moved in a vertical plane with 18 kg weight and 0.7m height [3]. The height of a pneumatic monopod robot was controlled with the PD controller, reported in [33]. Terence and team designed a 5-cm cube monopod-hopping robot with both static and dynamic stability [94]. The bio-inspired muscular-skeleton monopod robot was developed with pneumatic artificial muscles by mimicking human's leg [37], as shown in Fig. 2.1. PONGBOT-LEG equipped with high-torque motors with 2-DOFs is presented in [4], and it was planned to be developed further as a quadruped robot. The parallel-elastic actuation based monopod robot was developed to enable a large payload running [32]. The advantage of the monopod robots is the ability to can jump in a small area with no consideration of static stability. However, the disadvantage of them is the complex control strategy. The controller need to be well-design to suit their dynamic motion.

Biped Robots are the robots having two legs for locomotion. For example, the most remarkable one is ASIMO from Honda [36]. Nowadays, there are many biped robots developed mostly in humanoid shape. HRP-2Kai was developed by

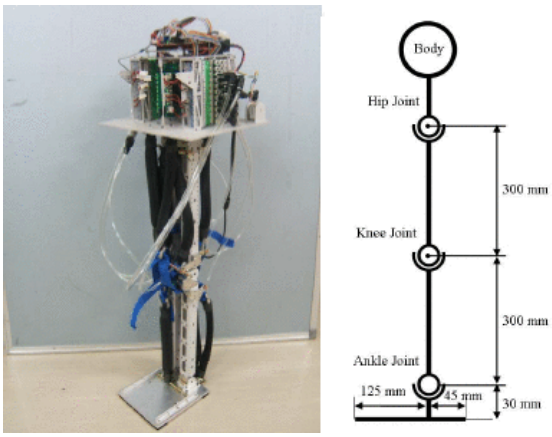


Figure 2.1: Bio-inspired muscular-skeleton monopod robot [37].

Humanoid Robotics Project in Japan for disaster response tasks [43]. There was total 30-DOFs with 6-DOFs for each leg. It was designed to be able to walk on uneven terrain, as shown in Fig. 2.2. The 41-DOFs KHR-3(HUBO) was presented in [61]. It operated locomotion with PD controller with many feedbacks, such as vibration reduction, position, and landing detection. In [79], the semi-passive biped robot was developed with five links of torso and two legs connected with two active hip joints and two passive knee joints, respectively. Fei Sun and team created a biped robot with 12-DOFs and analyzed the kinematic of the proposed robot [77]. They designed each robot leg following human mechanism, which are a hip joint, a thigh, a knee joint, an ankle joint, and a foot. As the humanoid robots, they can be operated in several applications, such as service robot, military robot, etc. In order to decide a biped robot, it need to be considered that the dynamic stability is also crucial to control the robot.

Multi-legged robots are the robots having more than two legs as the locomotion system, for example, tripod robots, quadruped robots, and hexapod robots. Recently, multi-legged robots are popular in research area, because they can be used in several applications in which walking on an uneven surface is required. Accordingly, they have the potential to walk on many kinds of terrains compared to wheel and track robots [50]. STriDER was a tripod robot designed with 4-DOFs for each leg with 3 joints at a hip and 1 joint at a knee [35], as illustrated in Fig. 2.3. In [92], the LARM tripod leg mechanism was proposed with 5 degrees of freedom for each limb. The soft membrane actuated robot with three legs was presented in [44]. As a result, it used vibration mechanism in order to move. Note

Figure 2.2: Walking biped robot on uneven terrain [43].

that this robot is quite different from the aforementioned ones as having only one actuator for each leg. Quadruped is a legged robot that has four legs. Figure 2.4 shows an example of the quadruped robot named JINPOONG [46]. Generally, quadruped robots are inspired by four-leg mammal, for instant, a dog (e.g. AIBO from Sony [25]). Due to existing works, the electric, hydraulic, and pneumatic actuators are mostly employed for the quadruped robots locomotion. For each leg, there are three joints imitating hip, knee, and ankle joints of mammal. A midsized dog was inspired to build the robot in [62]. It was equipped with pneumatic cylinders to mimic a muscle and a group of mercury switches to measure balancing of the body. The 12-DOFs quadruped robot, having 3 joints for each leg, was developed in [17]. The accelerometer was built-in the body to measure the speed as roll or pitch movement as a disturbance to control the robot. Along with the quadruped robot published in [54], it is a horse-inspired robot with 12-DOFs. However, a few numbers of DOFs based robot was discovered as well, with the concept of Spring Loaded Inverted Pendulum (SLIP) model. In [71], one degree of freedom leg was analyze dynamically with quadruped robot as well as two degree of freedom leg presented by [58]. In addition, the reptile-mimicked quadruped robots were reported in [24, 45, 60]. Apart from four legs, there are six-leg robots

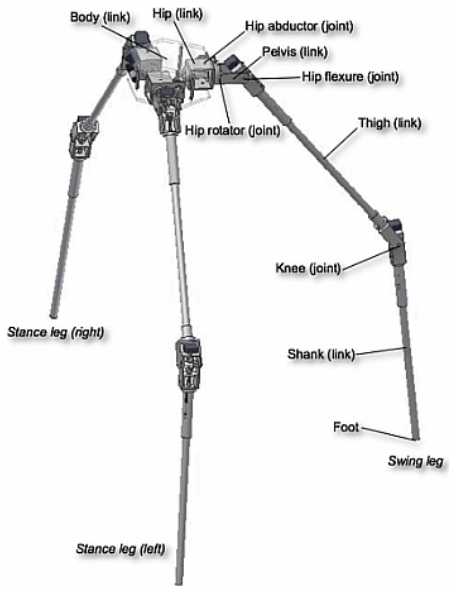


Figure 2.3: Self-excited tripedal dynamic experimental robot (STriDER) [35].

which called hexapod robots. For example, the RFDHR was developed 4-DOFs per one leg, and 1-DOF is a planar gear that let the robot be able to adjust position of legs [12]. Tedeschi and Carbone [81] summarized type of hexapod robot design as illustrated in Fig 2.5.

According to the literature review, there are several types of legged robots with different shape and structure. The advantage if legged robots is that they can perform in an uneven terrain. However, the legged robots are nonfunctional after getting damaged since the prior control strategies cannot be employed to operate efficiently with transferred models. Recently, several algorithms are published to overcome this problem. For example, the self-recovery algorithm that can assist the legged robot to move while damaged is proposed in [49]. However, it can be noticed that not only the algorithm, but also the structure of robot itself can make the robot become movable after getting damaged. In [14], the model analysis of different type of hexapod robot were report that the hexagonal model was better in turning and stability margin. Moreover, a novel design of hexapod robot that legs could rearrange to overcome the leg failures was proposed [105]. This paper proposes the novel structure of quadruped robot's legs based on the behavior of a caterpillar. The caterpillar-like robots have been developed so far as they cannot be used only for crawling but also climbing. The caterpillar employs a number

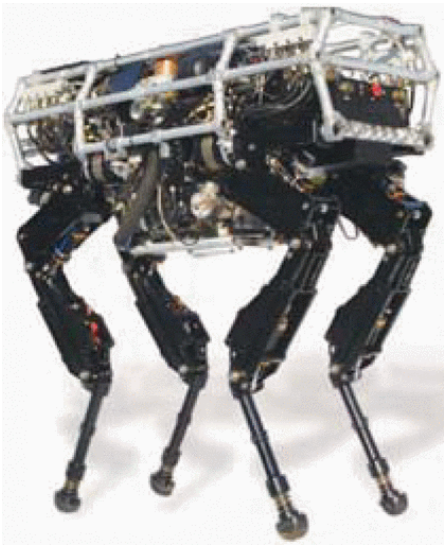


Figure 2.4: Example of a quadruped robot [46].

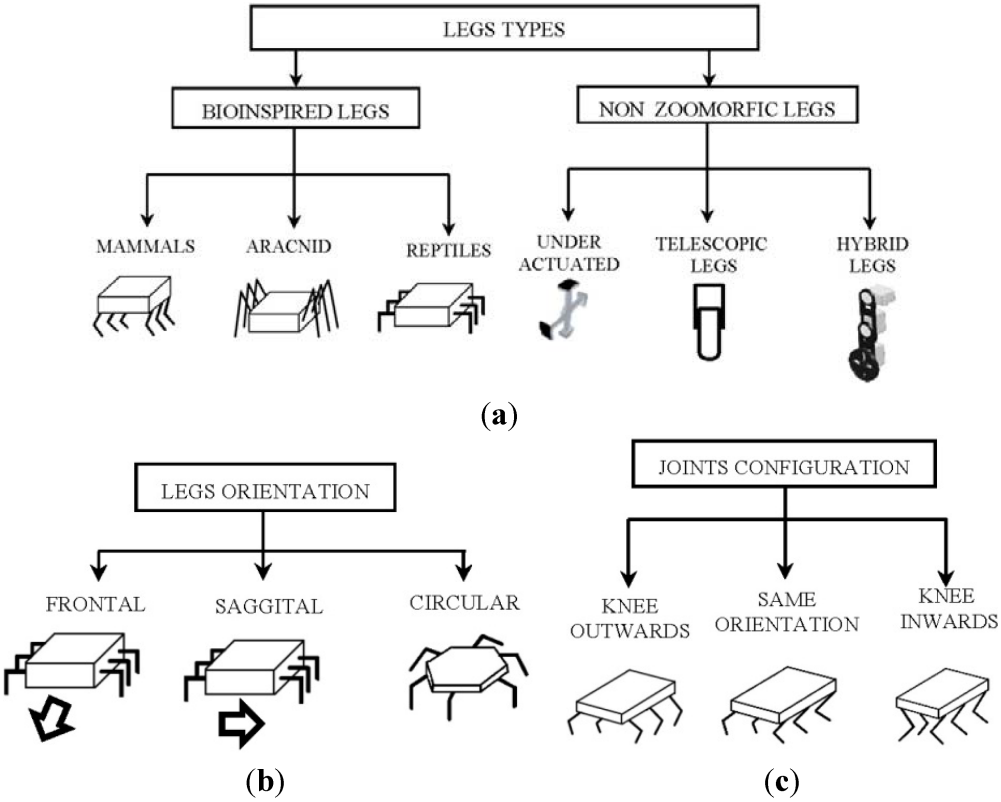


Figure 2.5: Types of hexapod leg's design [81].

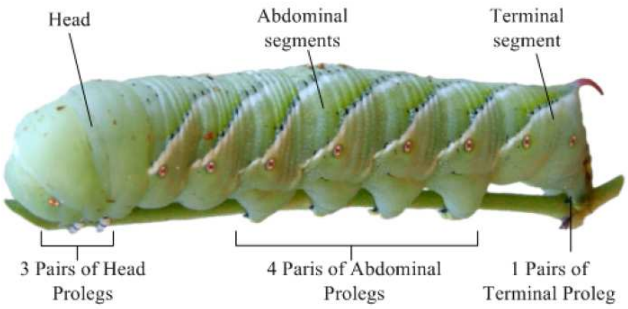


Figure 2.6: Caterpillar anatomy [90].

of prolegs, shown in Fig. 2.6, in order to perform crawling. As the caterpillar can move forward using the prolegs and the movement of body, this concept is applied to a quadruped robot to travel by using only one leg in case of damage. However, the structure of legs have to be designed circumspectly due to the face that the proleg can limit the reachable space of robot's leg while operating with normal quadruped gaits. In this paper, the new shape of robotic leg is designed with inspiration of caterpillar and optimized using PSO algorithm. The fitness function of PSO is set as the distance that robot can travel in both crawling behavior and trotting gait. The proposed robotic platform is called as 'Caterpillar-inspired Quadruped Robot' or CIQR. It is noted that this design process and performance test are conducted in simulation. The parts of the CIQR are printed with the 3D printer, and the performance test of the proposed quadruped platform is conducted as well.

2.2 System Description and Robot Model

In this study, the new quadruped robot is developed to increase the maneuverability of the ordinary legged robot after getting damaged. The reason why the four-leg robot is selected to develop is that, recently, several self-recovery algorithms were successfully investigated and performed with the robot that have at least 6 legs. However, most of the test cases are considered with 2 legs lost as a maximum number of broken legs, meaning that the robots still have 4 leg to compensate the movement. Dealing with the legged robot that has a small number of leg is mainly focused in this study. Figure 2.7 illustrates an example of quadruped robot model considered in this study. The developed quadruped robot consists of

4 legs and each leg contain 3 links so that it has 12 degrees of freedom.

2.2.1 Forward Kinematic of Robot Leg

The forward kinematic of robot leg is analyzed by using Denavit-Hartenberg parameters (also called DH parameters)[28]. It is the well-known method that is used to described the robotic system and other mechanical problem, for example, manipulator [5]. In DH parameters, there are four parameters defined as transformation parameters which are d_i, θ_i, a_i and α_i. The notation of each parameter can be described as follows [75]:

- d_i (joint displacement): the length between two joints.
- θ_i (joint angle): the angle measured between the orthogonal of the common normals. This parameter is variable for the revolute joint while the other parameters are constant.
- a_i (link length): the mathematical length of link (distance between the common normals).
- α_i (link twist): the angle measured between the orthogonal of the joint axes. For a prismatic joint, other parameters are fixed but this parameter is variable.

The top and side views of geometrical model of robot leg are shown in Fig. 2.8. As aforementioned, there are three links for each leg as link1, link2 and link3. The joint 1 will rotate in horizontal direction, and the joint 2 and 3 will move the leg up and down – vertical direction. The joint angles θ_1, θ_2 and θ_3 are the angles of link1, link2 and link3, sequentially. According to the parameters illustrated in Fig. 2.8, we can define the DH parameters of the systems as written in the Table 2.1.

Table 2.1: DH parameter for robot leg.

Link	d_i	θ_i	a_i	α_i
1	0	θ_1	a_1	90
2	0	θ_2	a_2	0
3	0	θ_3	a_3	0

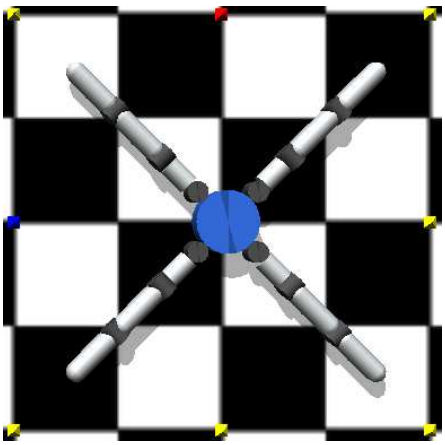


Figure 2.7: Quadruped robot model used in this study.

The homogenous transformation matrix H can be described by

$$H = \begin{bmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

$$H = T_n^0 = A_1 \cdots A_n \tag{2.2}$$

where R_n^0 is the 3×3 rotation matrix, and o_n^0 is the position of the end-effector respect to the base frame. Each homogeneous transformation A_i is given by

$$A_i = \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \end{bmatrix} \tag{2.3}$$

Then the homogeneous transformation of each joint can be derived as:

$$A_1^0 = \begin{bmatrix} c1 & 0 & s1 & a_1c1 \\ s1 & 0 & -c1 & a_1s1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$A_2^1 = \begin{bmatrix} c2 & -s2 & 0 & a_2c2 \\ s2 & c2 & 0 & a_2s2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

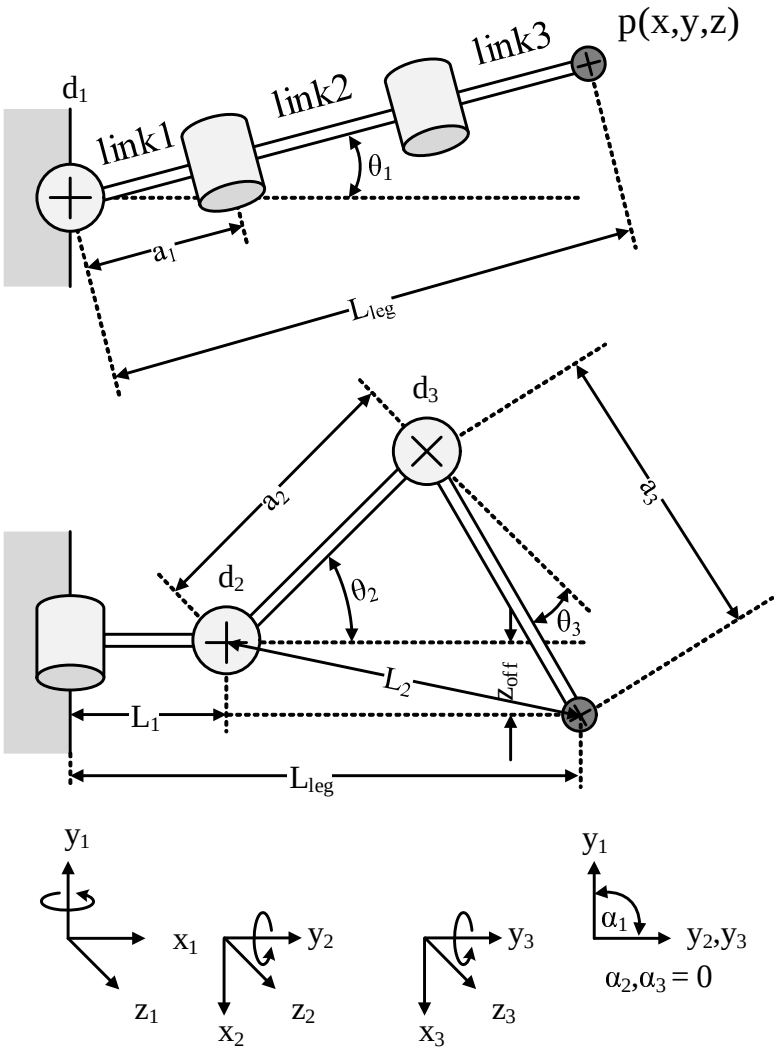


Figure 2.8: Geometrical model of robot leg.

$$A_2^3 = \begin{bmatrix} c3 & -s3 & 0 & a_3c3 \\ s3 & c3 & 0 & a_3s3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

where A_1^0, A_2^1 and A_2^3 are the homogeneous transformations of link1, link2 and link3, respectively. It is noted that $s1$, $s2$ and $s3$ are $\sin(\theta_1)$, $\sin(\theta_2)$ and $\sin(\theta_3)$, severally. Similarly with cosine function, $c1$, $c2$ and $c3$ are $\cos(\theta_1)$, $\cos(\theta_2)$ and $\cos(\theta_3)$, in order. Moreover, $\sin(\theta_2 + \theta_3)$ is written as $s23$, and $\cos(\theta_2 + \theta_3)$ is defined as $c23$ as well. We can find the total transformation matrix T of robot leg by using Eq. (2.2). Hence, T can be archived as the given equation below.

$$T = A_0^3 = \begin{bmatrix} c1c23 & -c1s23 & s1 & c1(a_3c23 + a_2c2 + a_1) \\ s1c23 & -s1s23 & -c1 & s1(a_3c23 + a_2c2 + a_1) \\ s23 & c23 & 0 & a_3s23 + a_2s3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

From Eq. (2.7), the position of end-effector of robot leg can be written as follows:

$$x = a_3 \cdot \cos(\theta_1)\cos(\theta_2 + \theta_3) + a_2 \cdot \cos(\theta_1)\cos(\theta_2) + a_1 \cdot \cos(\theta_1) \tag{2.8}$$

$$y = a_3 \cdot \sin(\theta_1)\cos(\theta_2 + \theta_3) + a_2 \cdot \sin(\theta_1)\cos(\theta_2) + a_1 \cdot \sin(\theta_1) \tag{2.9}$$

$$z = a_3 \cdot \sin(\theta_2 + \theta_3) + a_2 \cdot \sin(\theta_3) \tag{2.10}$$

2.2.2 Inverse Kinematic of Robot Leg

Forward kinematic, which is used to transfer the information of joint angel to the position of an end-effector, is explained in the previous subsection. Here, the solution of how to control the joint angles of robot leg to achieve the desired goal is written. In robotic and animation fields, this method is generally known as 'inverse kinematic'. There are several methods to get the inverse kinematic of the system, such as numerical calculation, Jacobian transpose method and geometric approach. In this study, the geometric approach is utilized to solve the inverse kinematic since there is only 3-DOFs of robot leg – simple to solve with this method. As shown in Fig. 2.8, we have L_{leg}, L_1 and L_2 as follows:

$$L_{leg} = \sqrt{x^2 + y^2} \tag{2.11}$$

$$L_1 = a_1 \cdot \cos(\theta_1) \tag{2.12}$$

$$L_2 = \sqrt{(L_{leg} - L_1)^2 + Z_{off}^2} \tag{2.13}$$

and the angle of the first joint can be described as

$$\theta_1 = \tan^{-1}(\frac{y}{x}) \tag{2.14}$$

and

$$\alpha_1 = \tan^{-1}(\frac{L_{leg} - L_1}{Z_{off}}) \tag{2.15}$$

$$\cos(\alpha_2) = \frac{a_2^2 + L_2^2 - a_3^2}{2 \cdot a_2 \cdot L_2} \tag{2.16}$$

$$\sin(\alpha_2) = \sqrt{1 - \cos^2(\alpha_2)} \tag{2.17}$$

$$\alpha_2 = \tan^{-1}(\frac{\sin(\alpha_2)}{\cos(\alpha_2)}) \tag{2.18}$$

so that we can get the θ_2 in Eq. (2.19).

$$\theta_2 = (\alpha_1 + \alpha_2) - 90 \tag{2.19}$$

The angle of the link3 can be calculated as follows:

$$\cos(\beta) = \frac{a_2^2 - L_2^2 + a_3^2}{2 \cdot a_2 \cdot a_3} \tag{2.20}$$

$$\sin(\beta) = \sqrt{1 - \cos^2(\beta)} \tag{2.21}$$

$$\beta = \tan^{-1}(\frac{\sin(\beta)}{\cos(\beta)}) \tag{2.22}$$

$$\theta_3 = 90 - \beta \tag{2.23}$$

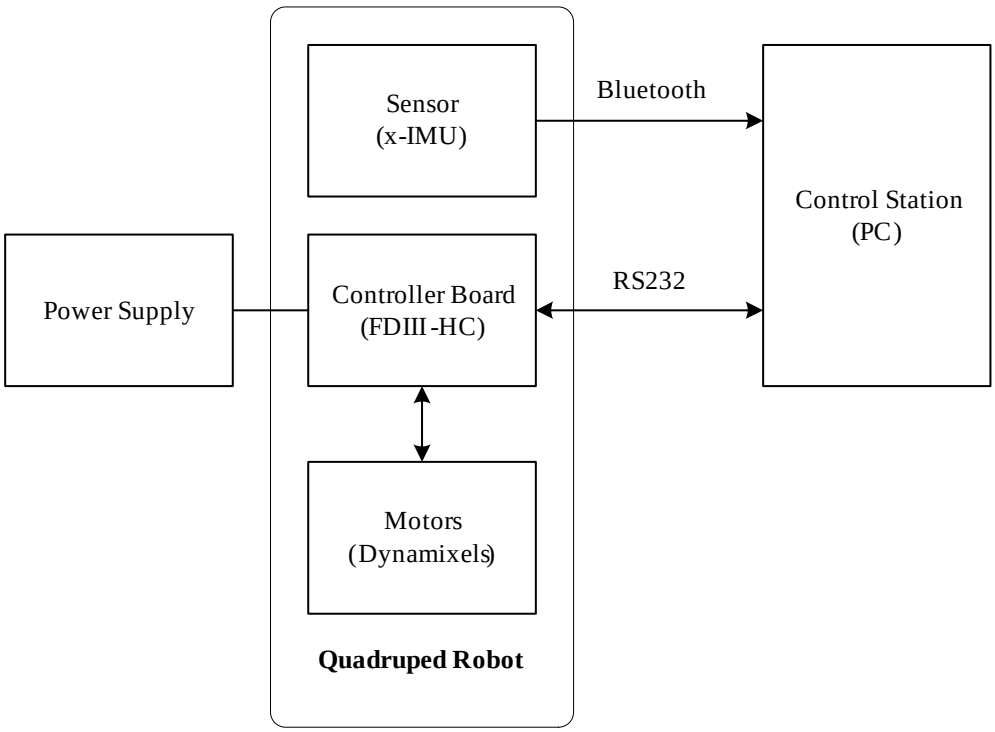


Figure 2.9: The block diagram of robotic system.

2.2.3 Robot Components

The design of the quadruped robot is geared towards the concept of modularity so that robot is developed to be assembled without difficulty. There are 6 main components of the proposed robotic platform, i.e. actuator, controller board, power supply, sensor and control station. Figure 2.9 illustrates the block diagram of overall system and details the connection between each component. As can be seen, there are three components mounted on the robot body, which are Dynamixel motors, xIMU and controller board.

Actuator

Among other equipments in robotic system, actuators are significant on account of the fact that the robot cannot function successfully without operating portion. Dynamixel motors, developed by Robotis company, were selected to act as the movement parts of the robot. Dynamixel is a digital servo motor used especially for robotic applications. It consists of direct current (DC) motor, gears and con-

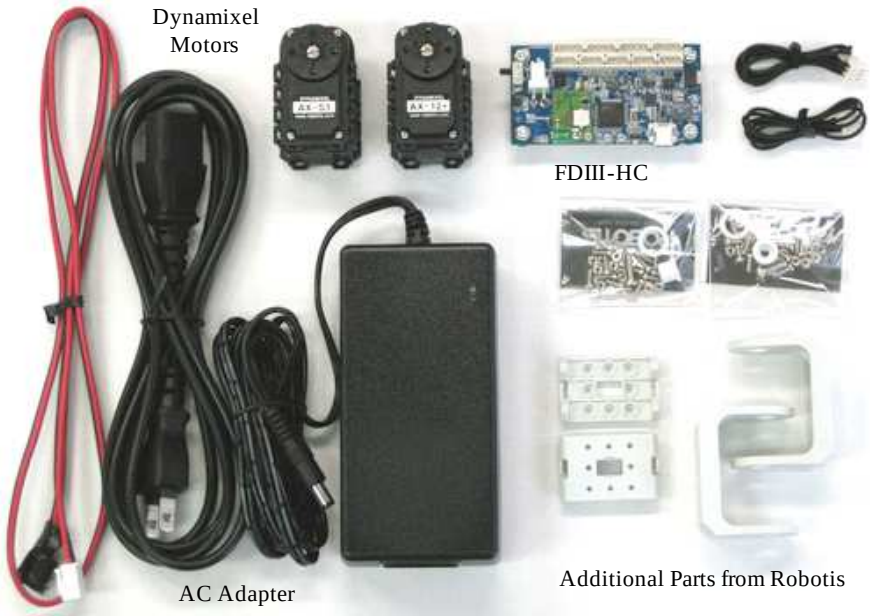


Figure 2.10: Components of quadruped robot (source:http://www.besttechnology.co.jp/).

troller which can be operated as a network system. We selected AX-12A of the Dynamixel series because of high speed and torque which is suitable for small robots. The aim of this study is to find to way to recover from getting damaged. As a result, we have built a small robot, at this point, to verify our hypothesis and algorithms. The size of the robot can be scaled up further to be able to operate in more practical situation. The example of the Dynamixel motor can be seen in Fig. 2.10.

Controller Board

Controller board is one of the most crucial part of the system. As shown in Fig. 2.9, the command will be transmitted to the controller board first in order to perform the movement of the robot. Given that we employ the Dynamixel from Robotis company which can be controlled and communicated with specific protocol, the FDIII-HC provided by BestTechnology was chosen to allow for a simplicity of programming.

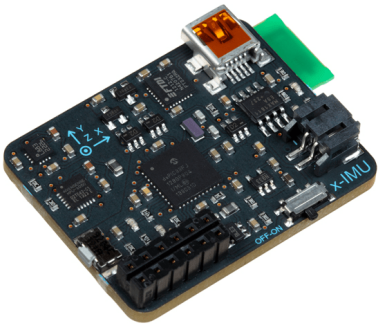


Figure 2.11: The x-IMU board (Source: http://x-io.co.uk/x-imu/).

Power Supply

For the power source, we used the AC adapter that come all together with the FDIII-HC controller board. Since this study focuses on the recovery methods for legged robots, the wired-type power source was chosen because it is one of the most rapid, simple ways to conduct the experiments. The robot utilizes it as the main source to drive the actuators. The photo of the AC adapter can be seen in Fig.2.10.

Sensor

According to the purpose of this study that the robot can utilize only internal feedback, the touch sensors – typically integrated with legged robots, is inadequate when performing with broken legs. Thus, the inertial measurement unit (IMU) is selected to provide the useful information, such as acceleration, orientation and direction. We chose the high-performance, well-calibrated IMU from the x-io Technologies company, named as x-IMU. The x-IMU has accelerometer, gyroscope, magnetometer, on-board SD card, battery and real-time clock, noticing that it can be used to provide more information as well, for example, position and velocity (by integrating the acceleration). In addition, it is able to be connected through the computer via both USB cable and bluetooth. In this study, the data are transmitted via the bluetooth by wireless technology. The photo of x-IMU can be seen in Fig. 2.11.

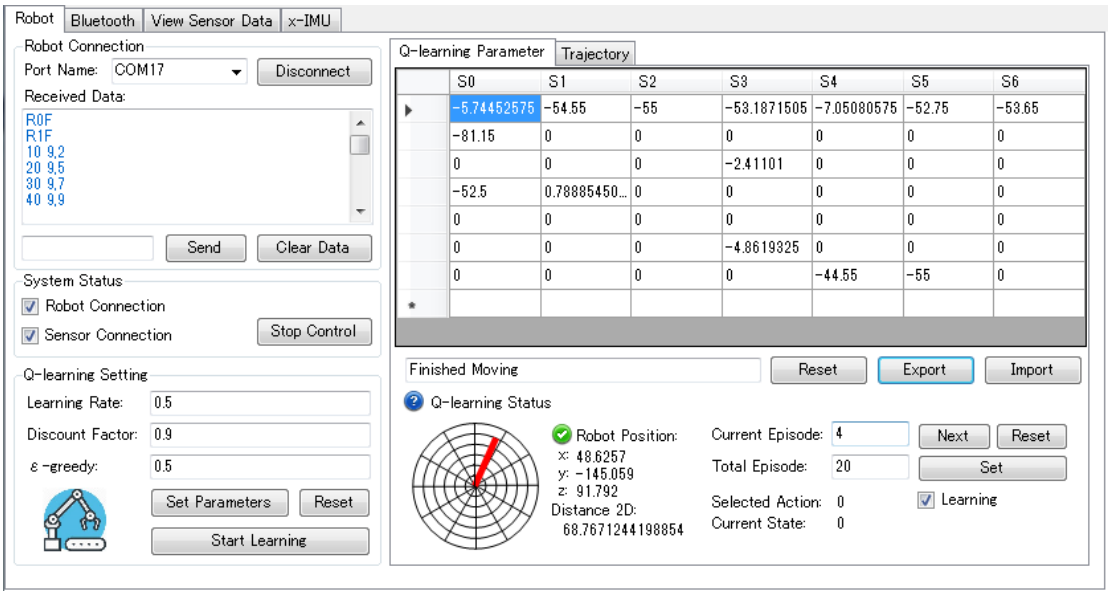


Figure 2.12: The GUI of main control station on the computer.

Control Station

In order to control the robot, we created a program to integrate the information received from x-IMU and the control signal sent to microcomputer. Since the self-recovery algorithm required the travelling distance, the short-time distance measurement is also processed and calculated using the acceleration data. This module is not only used for combining the information but also for deciding robot actions. The example of GUI interface is illustrated in Fig. 2.12.

Robot Body

In this study, we employ the 3D printing technology to make a prototype of the robot parts since it is more accurate and convenient to produce compared to making robot parts manually. The PLC filament is selected as the material supply of 3D printers due to its characteristics. It is beneficial for the components that will be assemble together. Fig.2.13 shows the sample of robotic parts printed by 3D printer. Moreover, the original parts provided by Robotis company are integrated on the robot body as well.

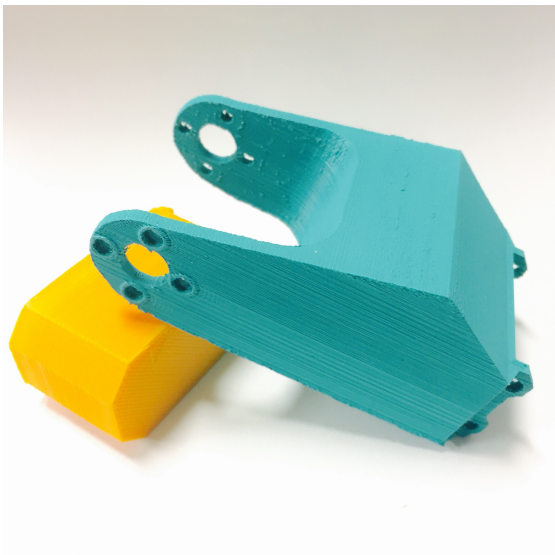


Figure 2.13: Robot leg parts printing by 3D printer.

2.3 Caterpillar-inspired Structure

In the nature, it is clearly seen that the animal can adapt itself extraordinary to survive on the earth. The flexibility is the main key to live and to reproduce. We have the inspiration that the bio-inspired mechanism can assist the robotic system to be more robust. According to [84], caterpillars can excellently perform climbing with their body and prolegs, providing fault-tolerant maneuverability. As a result, the caterpillar-inspired robot have been researched and developed to utilize in many application, for instant, wall-climbing [95]. As prolegs and the movement of body can be excuted to travel forward, this concept is applied to a quadruped robot to travel by using only one leg in case of damage. In this study, the design of the CIQR is geared towards the structure of caterpillar's prolegs. Each limb of robot is created as triangular shape to imitate the caterpillar's prolegs. The parameter l is the distance from the beginning of the limb to the foot of the altitude, and, h is the altitude (height) of the triangle. The quadruped robot used in this study has 3-DOFs per leg. Additionally, the prolegs are added only on the upper (b) and lower (c) limbs, which can be seen in Fig. 2.14. The reason of using triangle as proleg is that triangle can make the robot motion smoothly.

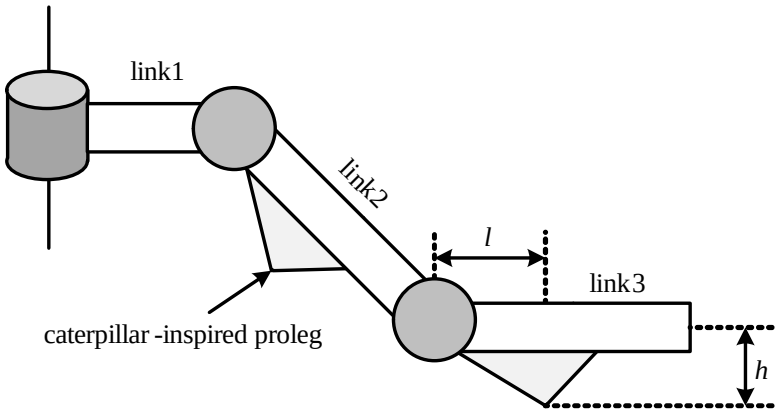


Figure 2.14: Leg structure of quadruped robot.

2.4 Optimization of Robot Structure

To ensure that the CIQR can perform well with both normal quadruped gait and caterpillar-inspired crawling behavior, the robot structure was optimized by operating with two types of locomotion. The first type of locomotion is trotting gait to control the robot because it is the fastest gait as reported by Darici et. al. in [16]. Two pairs of diagonal legs are moved back and forth between two states, as shown in Fig. 2.15. For the second type of locomotion, a sinusoidal generator is used to produce the rhythmic motion of caterpillar-like locomotion (crawling behavior). This method can ensure smooth motion and easy control over the motion [104]. The joint rotating angle can be calculated as follows:

$$y_i = Amp_i \sin(\frac{2\pi}{T}t + \phi_i) + O_i \tag{2.24}$$

where y_i is the rotation angle of joint i, Amp_i the amplitude, T the control period, t the time, ϕ_i the phase, and O_i the initial offset. Because the limb of robot is designed to be united, two parameters must be optimized, as expressed by Eq.(2.25),

$$P = \{l, h\} \tag{2.25}$$

where l and h are the parameters of the prolegs of link2 and link3, as illustrated in Fig. 2.14. The quadruped robot is programmed to execute the crawling behavior and the trotting gaits to ensure that the designed model can move properly

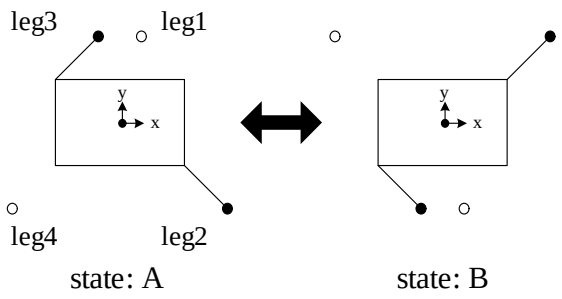


Figure 2.15: Trotting gait.

when crawling using one leg and when performing the normal quadruped gait. Therefore, the fitness function is set as the traveling distance, as follows:

$$D = \mu_1 \cdot d_c + \mu_2 \cdot d_t \tag{2.26}$$

where μ_1 and μ_2 are weight parameters and d_c and d_t are the distance traveled by robot with caterpillar-crawling behavior and trotting gait, respectively.

2.5 Results and Discussion

To achieve the designed upright structure, the shape of the legs of CIQR was optimized by numerical simulation as mentioned in the first part of this experiment. Thereafter the advantages of new designed structure were analyzed by conducting two sub-experiments, namely, recovering a robot by using conventional recovery methods in a simulation and operating the proposed robot with caterpillar behavior in a real application. Note that in both sub-experiments, the common structure and the proposed CIQR structure were compared.

2.5.1 Optimization of Robotic Structure

The simulation was run for 20 iterations with μ_1 and μ_2 as 0.5. The weight parameters were set to equal values because the CIQR is required to assign equal weights to both actions. Figure 2.16 shows the fitness values of optimization with time, and it illustrates that the robot can discover a new structure that can help it walk longer. The evolution of the robot leg can be seen in Table 2.2. At the beginning, the high prolegs cause the robot to walk slowly in accordance with

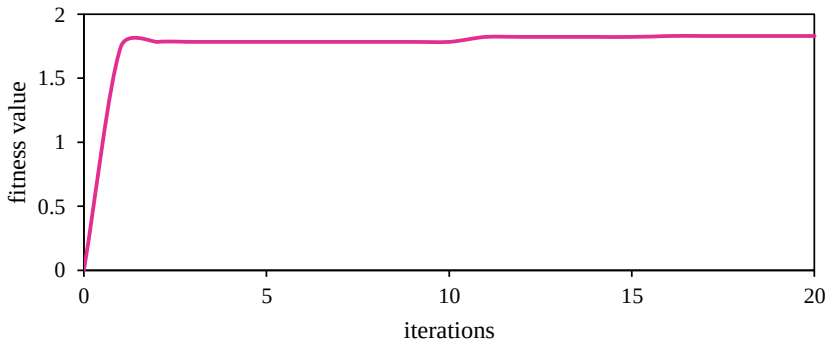


Figure 2.16: Fitness function of optimization process with PSO algorithm.

Table 2.2: Result of leg structure optimization with different numbers of iterations.

iteration	l (cm)	h (cm)
0	4.50	3.50
5	3.67	3.38
10	3.67	3.38
15	5.41	3.46
20	5.47	3.33

the fitness values shown in Fig. 2.16. After 5 iterations, the robot evolves to be suitable for crawling and trotting, such that the robot structure changes, and the fitness value is increased. In iteration = 20, the robot structure changes slightly. A comparison between the proposed model and the conventional model was made in this study. The results show that the optimized model can travel 39.19 cm and 13.34 cm in the trotting gait and crawling behavior, respectively. By contract, the normal structure can move 36.78 cm and 13.83 cm in the trotting gait and crawling behavior, respectively. Figure 2.17 shows the actual CIQR model fabricated using a 3D printer.

2.5.2 Simulation Experiments of CIQR with Conventional Recovery Method

In the experiment, the damage recovery algorithm was employed to compare the performance of the normal design (quadruped robot without prolegs) and the pro-

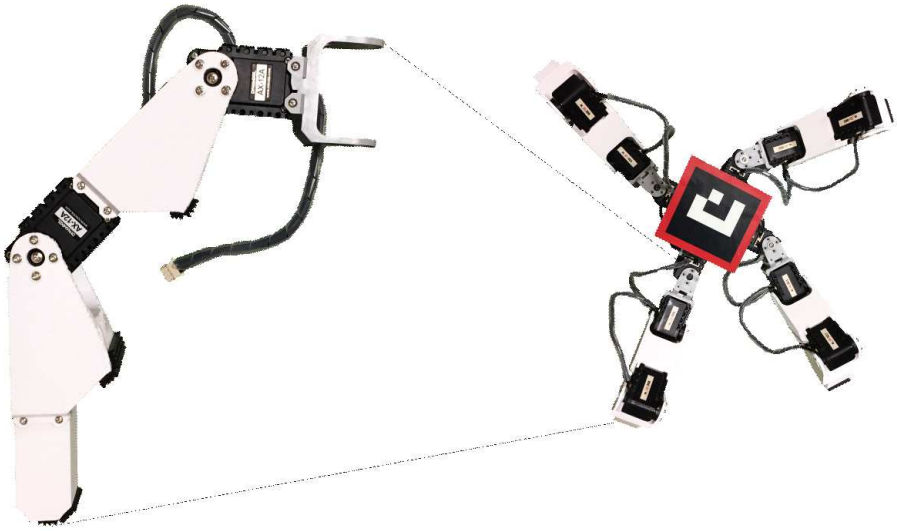
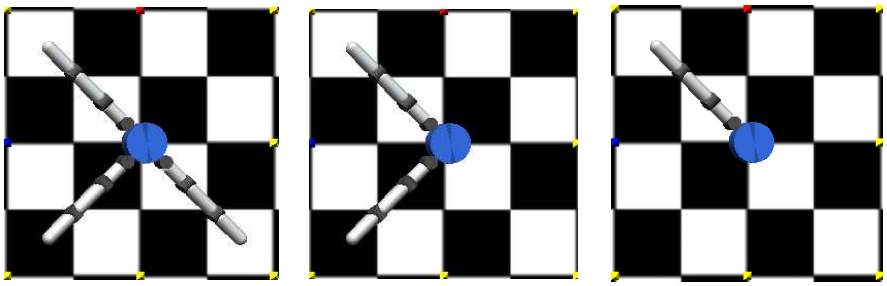


Figure 2.17: Proposed robot (CIQR).

posed design. The evolutionary adaptive gait, which involves creating sequential actions, was employed in this test; the concept underlying this process was inspired by existing works presented in [63] and [26] as described in section 4.2. Three experimental scenarios were tested in this study, namely, one leg loss, two leg loss, and three leg loss, as shown in Fig. 2.18. A simple algorithm was employed to control the damaged robot. For each joint, five rotational angles are controlled in sequences. The robot performed motions step-by-step in round-robin fashion, from the first angle to the last one. Because CIQR has 12 joints, 60 parameters in total are used to control the robot. To determine 60 parameters, PSO was used to execute the discovery process once again [31]. At the beginning, all parameters were set randomly. Next, the robot performed the first action with the first rotational angle of each joint. At $t = T$, the robot performed the second action. This step was iterated until t reached the setting time. The fitness function of this algorithm was set as the distance that the damaged robot could travel. The test results obtained with the damaged robot are given in Table 2.3. As can be seen, with only one leg lost, CIQR performed better than the normal robot in terms of the total distance traveled. By contrast, the normal robot produced a good

Table 2.3: Distance traveled by damaged robot.

Number of legs lost	Normal robot (cm)	CIQR (cm)
1	48.68	51.67
2	92.42	85.41
3	17.08	131.73



1 leg loss 2 leg loss 3 leg loss

Figure 2.18: Test cases of damaged robot.

result in the second case as it walk about 10 cm more than the proposed robot. The CIQR moved slower in this case possibly because of the additional weight of the prolegs. In case of only one leg, CIQR could travel longer than the normal robot. However, the control method used in this benchmark could not efficiently control the robot because the robot motion was not smooth, and the possibility of the robot flipping over during the evolutionary process prevailed. These problems were tested once again using the proposed recovery algorithm (i.e. SLMIC) described in section 4.3.

2.5.3 Experimental Results with Caterpillar-inspired Crawling Behavior

To ensure that the proposed robot could function in real scenarios, an experiment with the actual robot was conducted. In this case, the performance of CIQR was compared with that of the normal robot. The evaluation was conducted with a damaged robot having one functional leg. As in the simulation, both robots were programed to move forward according to Eq. (2.24) with the same set of parameters, that is, $A_2 = A_3 = 40$, $\phi_2 = 0$ and $\phi_3 = 30$, which are the

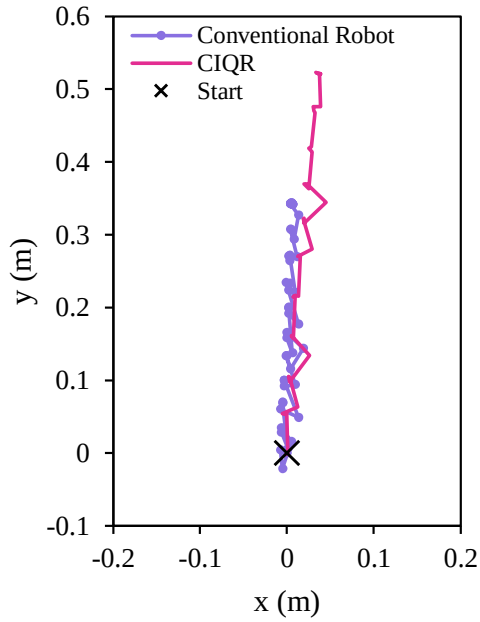


Figure 2.19: Comparison of trajectories traveled by conventional robot and CIQR-based robot with three legs lost.

amplitudes for controlling link2 and link3, and the initial offsets of link2 and link3, respectively. Link1 was set to the fixed direction of 0° to ensure the robot traveled straight. The results show that both robots could travel straight but shifted slightly toward the right. However, the novel structure of CIQR traveled longer than the normal robot in the simulation, as shown in Fig. 2.19. The normal robot moved forward and stopped after 20 s, traveling 34.17 cm in the course. In the same period, the CIQR traveled 52.08 cm. Additionally, the positions of both robots in the experiment were measured using an overhead camera and image processing technique.

2.6 Summary

This paper proposes the new structure of a quadruped robot, so-called as CIQR, inspired by the caterpillar's proleg. The legs of the robot are designed with the triangular shape optimized by the PSO algorithm. The distances traveled by the robot (trotting gait and crawling behavior) are set as the fitness function of

optimization to ensure that the proposed robot structure can operate with both normal quadruped gait and crawling behavior properly. The results show that the proposed method can make the robot move with a single leg, but the robot with normal structure cannot travel well in both simulations and experiments. Moreover, the CIQR was tested with the recovery method. The results show that CIQR can perform self-recovery after getting damaged.

Chapter 3
Damage Detection Algorithms

3.1 Introduction

Fault detection is crucial for mobile robots for improving robustness to ensure that no damaged is caused to robots and environment when losing control [86, 87]. In [101], they categorized the robotic faults into three main faults, i.e. sensor faults, actuator faults and system faults. Generally, fault detection techniques have been researched in several fields of robotics. For example, Zhou Gao and team developed the wheel-legged robot to cope displacement sensor fault with deep learning [27]. Manipulator fault detection techniques were studied in [59, 85]. The fault detection method based on sensor fusion for navigation and obstacle avoidance tasks was tested successfully with redundant sensors [2]. Torque prediction error was developed in [106] to help modular robot detect the fault occurred in the system. The support vector data description (SVDD) was used with internal sensors to detect fault of mobile a robot [21]. In [20], the particle filters was employed for estimation the fault. A combined logistic and model based approach was proposed to detect the fault of the climbing robot in [102].

Leg-loss identification is a significant process to assist the damaged robots to discover the new, present models. The camera equipped on the robot is employed to detect the artificial markers mounted on robot's legs [70]. This method works suitably to detect the abnormal actions of leg loss; yet, there are some conditions of using camera and image processing that lead to task failures. For example, intensity of light or dusty environment can affect a detection system. Although the artificial markers aid in recognizing position of legs faster and easier, predesigned markers are needed to attach on robot's legs in advance. Some researchers also detect the abnormal joints of legged robot by measuring the electrical current spent while operating [66]. The advantage of this method is that external sensing measurement is not employed which make it more flexible in practical situation.

However, the errors might occur with low battery as measuring electrical current incorrectly. Another method, named as virtual joint sensor, can detect the joint sensor fault on humanoid robot [34]. This method uses the linear invert pendulum model and the leg kinematic models cooperating with the Kalman filter to detect and recovery the fault of joint sensors. Although, the simulation results prove that this method can detect the fault of joint sensor effectively, it can detect only the fault of joint sensor. Acceleration data is applied to identify the fault as well [41]. It is used to measure the orientation of robot and set as the input of the state machine. This method can detect only the abnormal behaviors of the robot.

In spite of the fact that aforementioned methods can correctly detect the broken parts or abnormal behavior, the new-model of robot is not provided. Since the method proposed in [70] can identify only the status of each leg (usable or unusable) and the method proposed in [66] can detect the joint fault, they cannot provide actual updated model after damaged. Fault diagnosis is a crucial operation for mechanical systems to prevent from unexpected actions caused by failures. Generally, additional sensors (e.g., current sensors) are integrated to the system to inspect the abnormal behaviors – for instant, unusual load current or load torque. However, in order to get the system response, certain types of sensors are required to install inside the systems, such as a tachogenerator, which can lead to minor losses. To overcome this issue, some researchers employ acoustic processing to determine the failures. It is beneficial that acoustic processing can operate independently from the main systems as the sensing devices are not necessary to attach on the systems, and therefore it is widely used in machine health monitoring system as well.

Typically, fault diagnosis of machinery systems is done by means of feature extraction and classification. In [80], sound of the system and responses of vibration sensors are combined in cooperation to detect the vibration of the system. The result shows that it can detect the vibration frequency correctly. Acoustic analysis is also applied with induction motor to identify the faults [68]. The Fast Fourier Transform (FFT) and Spectral Analysis are implemented to detect the bearing and unbalance faults successfully in the experiments. The faults of automobile engine are detected by using FFT and Correlation-based Feature Selection as feature extraction methods [29]. Support Vector Machine (SVM) is employed to perform the fault classification, subsequently. This approach can identify the faults correctly by 88 percent accuracy. Besides aforementioned extracting methods, the Wavelet Transform (WT), the well-known method for frequency and time

response, is implemented in [74]. The abnormality is distinguished effectively by using energy ratio of frequency-band. In [38], multi-class SVM is operated to classify the imperfection of the systems with wavelet packet entropy. In traditional techniques of classifying process, not only SVM acts as a classifier but other classical methods, i.e., Artificial Neural Network (ANN), are also used extensively to distinguish between two sets of behaviors – healthy and unhealthy. Certain types of ANN are further implemented successfully in [73, 93].

The PSO-based Leg-loss Identification method (PLI) is proposed in this chapter [10]. The PLI method uses only on-board sensors that lets robot become more versatile. Particle swarm optimization is utilized to optimize the fitness function that is set as the resemblance of candidate models and actual damaged robot. The advantage of PSO is that parameters can be set as the real number, meaning that decoding and encoding process (using in general Genetic algorithm) are not essential. Since using only final posture can cause the confusion of model identification, it is better to observe the behavior of robot during a period of time. The comparison between actual-damaged robot and candidate models in the simulation is done using normalized cross-correlation algorithm. The performance of PLI is evaluated using the Open Dynamics Engine™ (ODE) simulator. In the benchmark, the efficiency of PLI is compared with existing GA-based identification method [51]. The results show that the proposed method can assist legged robots to discover the new leg's lengths better than GA-based method in term of accuracy. Moreover, two experiments are conducted with 2-DOFs quadruped robot to apply this method in practical situation. According to the result, the PLI algorithm can provide the acceptable new-updated model so that the proposed method can employ with actual robot as well. However, the gap between real world and simulation slightly limit the performance of algorithm. Furthermore, the acoustic-based fault diagnosis for legged robots (AFL) is developed to detect the abnormalities of joints [9]. Sound of servo motors are recorded simultaneously while a walking legged robot is executed to perform specific actions. The recorded signals are normalized afterward before analyzing frequency response with FFT. The energy of each frequency band will be calculated for feature extraction process. For classifier, the fuzzy logic classifier (FLC) is selected to avoid a drawback of conservative method, such as ANN, which can be overfitting in complicated model. In addition, the results of the AFL method are compared with ANN-based approach, and they show that the proposed method can provide preferable outcome in term of accuracy. Lastly, it can be concluded that the AFL method

is feasible to utilize in the domain of fault detection on legged robots.

3.2 Leg Loss detection Algorithm

The PSO-based Leg-loss Identification method (PLI) is proposed in this paper. The PLI method uses only on-board sensors that lets robot become more versatile. Particle swarm optimization is utilized to optimize the fitness function that is set as the resemblance of candidate models and actual damaged robot. The advantage of PSO is that parameters can be set as the real number, meaning that decoding and encoding process (using in general Genetic algorithm) are not essential. Since using only final posture can cause the confusion of model identification, it is better to observe the behavior of robot during a period of time. The comparison between actual-damaged robot and candidate models in the simulation is done using normalized cross-correlation algorithm. The performance of PLI is evaluated using the Open Dynamics Engine™ (ODE) simulator. In the benchmark, the efficiency of PLI is compared with existing GA-based identification method [51]. The results show that the proposed method can assist legged robots to discover the new leg's lengths better than GA-based method in term of accuracy. Moreover, two experiments are conducted with 2-DOFs quadruped robot to apply this method in practical situation. According to the result, the PLI algorithm can provide the acceptable new-updated model so that the proposed method can employ with actual robot as well. However, the gap between real world and simulation slightly limit the performance of algorithm.

3.2.1 Robot Model and Simulation

The quadruped robot, consisting of twelve servo motors (three joints for each leg), is employed to evaluate the proposed algorithm in this paper. The attended mode of servo motors is shown in Fig. 3.1. There are two limbs for one leg, upper limb and lower limb. The lower limb and upper limb are connected together with the hinge joint, but the body and the upper limb are connected with two hinge join, as shown in Fig. 3.1(b) and (c). Both upper and lower limbs are constructed as the cylinders with 2 cm in radius and 20 cm in length. The shape of the robots' body is also in the cylinder shape with 30 cm diameter. To measure the orientation, used as the main cue for the PLI, the Inertial Measurement Unit (IMU) is mounted on the robot body. The Open Dynamic Engine™ (ODE) is utilized in order to

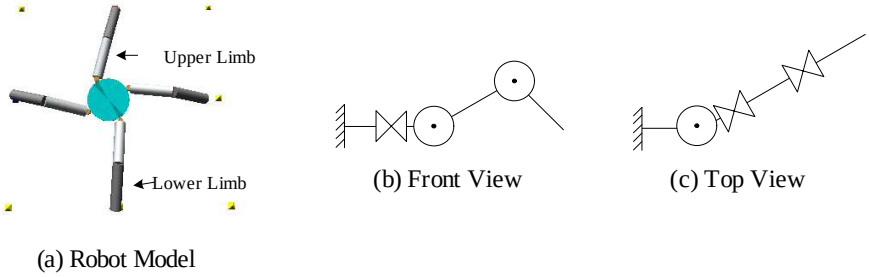


Figure 3.1: The robot model used in the simulation: (a) reference quadruped robot, (b) joints construction in front view and (c) joints construction in top view.

simulate and evaluate the performance of proposed method.

3.2.2 Methodology

Leg-loss identification method is beneficial for legged robots in order to deal with the damaged legs. This paper mainly focuses on identifying the length of broken legs. In the PLI method, the normalized cross correlation and PSO are employed to evaluate the differences between damaged robot and candidate model and to discover the candidate model providing similar behavior to actual damaged robot, respectively.

Normalized Cross-Correlation (NCC)

The cross correlation is utilized to compare the responses of damaged robot and candidate model. It is generally used in signal processing with the concept of convolution technique to compare between an input signal and a reference signal [76]. In PLI, the normalized cross-correlation is implemented as its result is allocated between +1 and -1. In case that the result is close to +1, it means that two signals are completely similar. On the other hands, if the result is close to -1, it means that two signal are exceedingly different. The cross-correlation between two real continuous signals x and y can be computed using Eq. (3.1),

$$C_{xy}(\tau) \equiv \int_{-\infty}^{\infty} x(t)y(t-\tau)dt \tag{3.1}$$

where the time shift τ is the *lag*. The discrete normalized cross-correlation of signal x and y over the period of time N is given as follows:

$$C_{norm} \equiv \frac{\sum_{n=0}^{N} x[n] \cdot y[n]}{\sqrt{\sum_{n=0}^{N} x^2[n] \cdot \sum_{n=0}^{N} y^2[n]}} \tag{3.2}$$

In PLI, as orientation is employed, there are three signals, i.e., pitch, roll and yaw angles of robot's body, to be considered. The NCC result is combined and averaged in one value as following equation:

$$cw = C_{norm-ave} = \frac{C_{pitch} + C_{roll} + C_{yaw}}{3} \tag{3.3}$$

where C_{pitch}, C_{roll} and C_{yaw} are the normalized cross-correlation of pitch, roll and yaw angles of robot's body, respectively. The value of cw can be varied from -1 to +1; hence, on condition that it allocates at 1, it means that the candidate model and broken robot have similar model (same behaviors). On the contrary, it indicates that candidate model and broken robot have different model when cw is close to -1. In addition, the value cw will be manipulated in PSO algorithm in next process.

Particle Swarm Optimization (PSO)

The PSO, inspired by swarm behavior of living creature, is generally used in optimization problems. The solution of the problem is called as 'particle'. The concept of PSO can be described that all particle in the search space will follower the leader (highest fitness value) to find optimum point. In the search space, there are three main parameters, which are current position (x), best previous position (p) and current velocity (v). For ith particle in a D-dimension space, it can be set as $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$, $P_i = (p_{i1}, p_{i2}, ..., p_{iD})$ and $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$. The best prior positions of any particles in population are stored for updating their speeds. The evolution of each particle is varied by the following equations:

$$v_{id} = w \cdot v_{id} + c_1 \cdot \text{rand}() \cdot (p_{id} - x_{id}) + c_2 \cdot \text{rand}() \cdot (p_{gi} - x_{id}) \tag{3.4}$$

and

$$x_{id} = x_{id} + v_{id} \tag{3.5}$$

where w is the weight to balance between global search and local search, c_1 and c_2 are the positive constant weights, and rand() is a random function range 0 to

1. The velocity of each particle is updated by combining best local solution (p_{id}) and best global solution (p_{gd}). Eq. (3.4) and Eq. (3.5) are originally published in [31]. In PLI, the Eq. (3.5) is modified to improve the robustness as parameter cw is added into the equation as follows:

$$x_{id} = x_{id} + (1 - cw) \cdot v_{id} \tag{3.6}$$

The term $(1 - cw)$ is contributed the system to become convergent rapidly. The term $(1 - cw)$ become 0 when cw is +1, meaning that the updated model of damage robot is achieved (a candidate model can produce same behaviors as broken robot). Thus, it is not in need of updating the position (x).

PSO-based Leg-loss Identification (PLI)

The PLI algorithm can be applied to assist the legged robot to identify the new updated model after damaged. This method is able to get the length of robot's legs by means of comparing the responses of an actual robot and candidate models in the simulation. The NCC and PSO are further employed in PLI algorithm to obtain the similarity of behaviors and to evolve the candidate robots in the search space, consequently. The advantage of this algorithm is that only internal sensors are utilized. Thus, it leads the legged robot become feasible to operate in unfamiliar environments. The procedure of PLI algorithm can be seen in Fig. 3.2. The algorithm begins with creating the N random particles in the population. Each particle contains eight real values – four legs (two parts per leg) – as the robot model illustrated in Fig. 3.1 (a). The value of particle (p), so-called as candidate model, of n can be represented as follows:

$$p_n = \{L_{n,1}, L_{n,2}, L_{n,3}, L_{n,4}, L_{n,5}, L_{n,6}, L_{n,7}, L_{n,8}\} \tag{3.7}$$

where L is the length of robot's legs and $n = 1, 2, 3, ..., N$. In primary state, it is assumed that only lower parts of legs are broken. The lower lengths of legs will be generated randomly with range of 0 to 60 cm, but the upper lengths will be set as x cm identically to the length of actual robot – the lower lengths should be constant since they have not been broken yet. After initializing the particles, all of candidate models and actual robot will perform similar action that is generated randomly beforehand. Later, the evaluation process begins. All of candidate models will be compared with damaged robot following Eqs. (3.1) and (3.2). If the best cw value of the candidate model reaches the desired value m (it is set

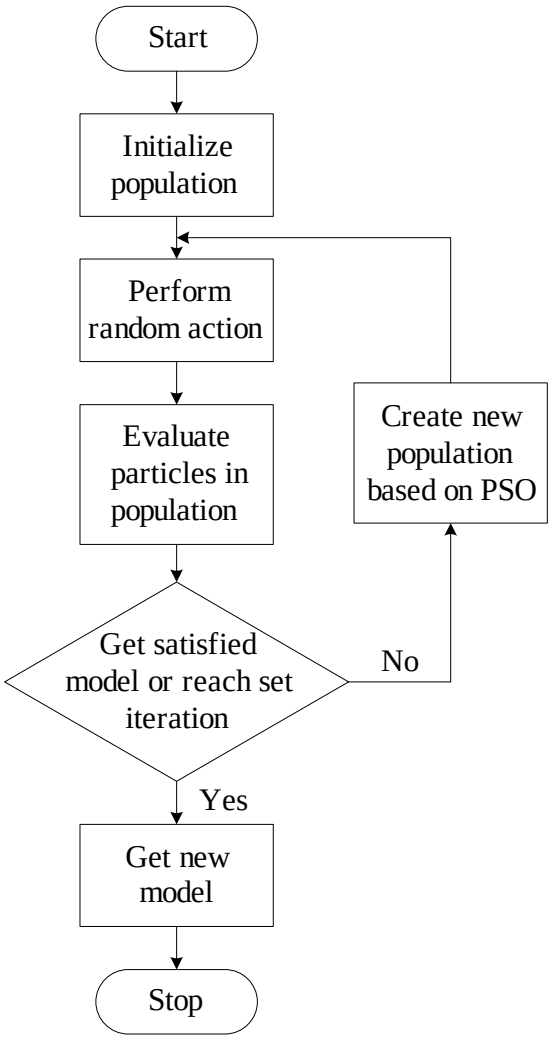


Figure 3.2: Process of PLI algorithm.

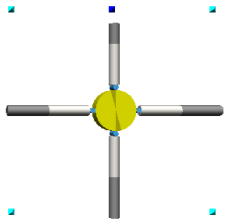
as 0.95 in this study), the process of leg-loss identification will be completed, and select the best model as the new model for damaged robot. Moreover, if the iteration reaches the time t, this process also ends. In case that the model is not satisfied, the new set of candidate model will be generated according to the PSO algorithm – all of L in p_n will be updated following Eqs. (3.4)-(3.5), and then continue the same process as shown in Fig. 3.2. This routine will be operated repeatedly until reaching the desired cw or set time t. Figure 3.3 illustrates the best candidate model in search space that can adapt itself to provide same behavior as damaged robot (the lengths of upper limbs and lower limbs are 20 cm but the length of broken limb is set as 10 cm). By performing for 25 iterations, the robot can discover the new-updated model. Additionally, in evolution process, some of L values in p_n will be varied randomly with probability $prob = 0.10$, and the upper length of legs will not be changed if the lower length of legs are not zero.

3.3 Joint Fault Diagnosis Algorithm

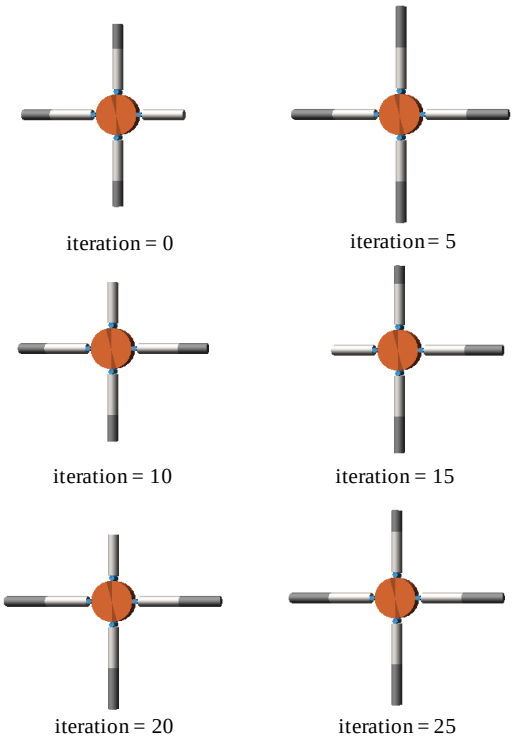
In this paper, the acoustic-based fault diagnosis for legged robots (AFL) is developed to detect the abnormalities of joints. Sound of servo motors are recorded simultaneously while a walking legged robot is executed to perform specific actions. The recorded signals are normalized afterward before analyzing frequency response with FFT. The energy of each frequency band will be calculated for feature extraction process. For classifier, the fuzzy logic classifier (FLC) is selected to avoid a drawback of conservative method, such as ANN, which can be overfitting in complicated model. In addition, the results of the AFL method are compared with ANN-based approach, and they show that the proposed method can provide preferable outcome in term of accuracy. Lastly, it can be concluded that the AFL method is feasible to utilize in the domain of fault detection on legged robots.

3.3.1 System Description

The legged walking robot used in this study consists of four legs with 2-DOFs per each leg as illustrated in Fig.3.4. Eight servo motors are built-in to act as actuator for each joint. The robot is connected directly to computer to execute the assigned action. In the experiments, the acoustic signals of the robot are recorded via the USB microphone allocating on the center of the robot, illustrated in Fig.



(a) Damaged Robot



(b) Evolution of the best candidate model in search space

Figure 3.3: The evolution of the best candidate robot in search space: (a) damaged robot and (b) the best candidate model captured every 5 iterations – shown that the candidate model can evolve itself in order to create the behaviour as same as damaged robot. In 20 iterations, it cannot find the proper model, but it can identify the length of legs correctly within 25 iterations.

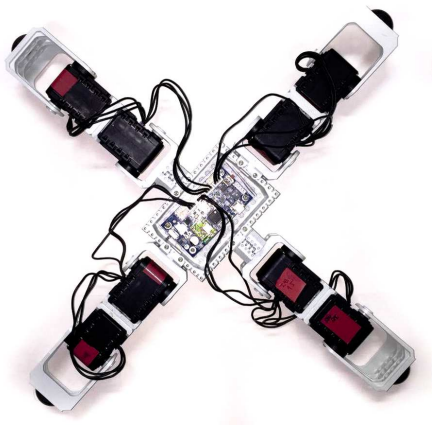


Figure 3.4: The quadruped robot used in the experiment and study.

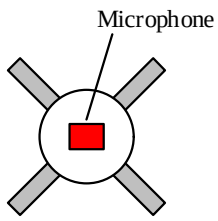


Figure 3.5: The position of microphone.

3.5. There are two types of joint diagnosis in this study, healthy and unhealthy joints. The healthy joints are defined as normal joints that can function properly in the practical situation; whereas, unhealthy joints are uncontrollable joints, i.e., locked joints, non-powered joints and broken-gear joints.

3.3.2 Theoretical Background

Fast Fourier Transform (FFT)

Fourier Transform (FT) is extensively used in digital signal processing to analyze the frequency response. In this study, the sound of motor is collected as a discrete signal; as a result, the FT is calculated by force of circumstance in discrete domain, called as Discrete Fourier Transform (DFT). The FFT is an efficient computation

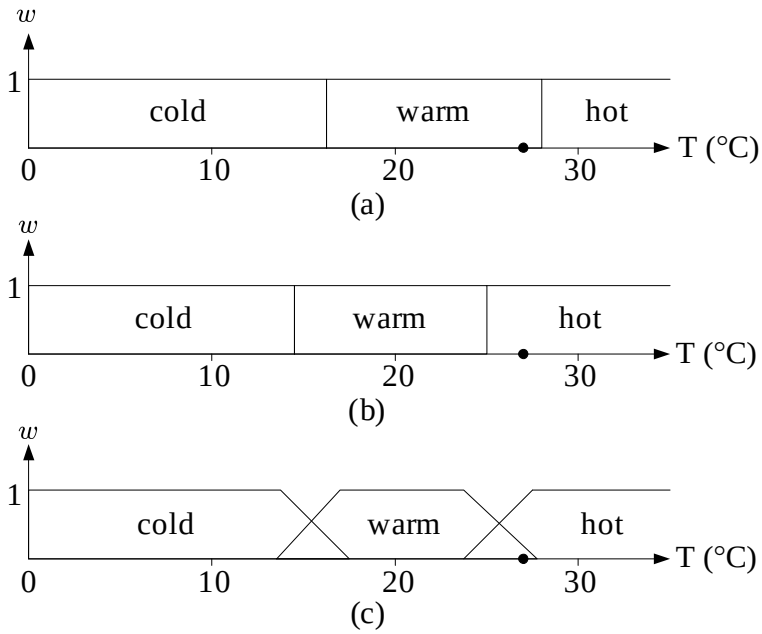


Figure 3.6: Example of non-fuzzy and fuzzy classification: (a) 27°C $\in$ *warm*, (b) 27°C $\in$ *hot* and (c) 27°C belongs to both *warm* and *hot* sets with $w = 0.2$ and $w = 0.8$, respectively.

of the DFT, namely

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \tag{3.8}$$

$$W_N = e^{-j2\pi/N}, \tag{3.9}$$

where $X(k)$ is the kth coefficient of a length N sequence $\{x(n)\}$ and $k = 0, ..., N-1$ [64]. The amplitude of frequency spectrum $X(k)$ can be calculate by

$$A(k) = \frac{|X(k)|}{N} = \frac{\sqrt{\mathrm{Re}(X(k)^2) + \mathrm{Im}(X(k)^2)}}{N}, \tag{3.10}$$

with corresponding frequency:

$$f(k) = \Delta f \cdot k, \tag{3.11}$$

where $\Delta f = f_s/N$ and f_s is the sampling frequency.

Fuzzy Logic Classifier (FLC)

In order to label the data, there are several types of classifiers that can be utilized to cluster the groups of data in different manners. The advantage of fuzzy classifier

is that it allows linguistic labels which is equivalent to what human used in real life, such as warmer, warm, cold, colder and etc., [39]. The dissimilarity between fuzzy classifier and non-fuzzy classifier is illustrated in Fig. 3.6. For non-fuzzy classifier, the data can be in either one set or the other, whereas it can belong to two sets in fuzzy classifier. For example, the 27°C can be labeled in only one set, *warm* or *hot*, for non-fuzzy classifier. On the other hand, the 27°C can be a member of both *warm* and *hot* sets but with different weights(w) as 27°C $\in$ *warm* with $w = 0.2$ and 27°C $\in$ *hot* with $w = 0.8$.

3.3.3 Methodology

The AFL method is divided into two processes, feature extraction and classification, conventional processes used in existing methods. The algorithm begins with recording the sounds of motors attached on the legged robot, while robot is performing specific actions – activating each joints individually. In the feature extraction, the acoustic signals will be transferred to the next processes, normalization, FFT and energy calculation, sequentially. The decision making of fault detection will be done afterward by using fuzzy classification. The overall processes of AFL algorithm is shown in Fig. 3.7.

3.3.4 Feature Extraction

After collecting acoustic data, the feature of each signal will be extracted using three main processes as follows:

Normalization:

In order to avoid the effect of various intensity of sound, all of the acoustic signals will be normalized by maximum peak – divided the waveform by the peak value. Fig.3.8 and Fig.3.9 illustrate the examples of normalization results of both healthy and unhealthy joint signals.

FFT:

As aforementioned, there are several ways to extract the feature of sound signals. The FFT is employed to perform this procedure in this study by the reason of sufficient information provided. The frequency responses of healthy and unhealthy (locked joint) signals are shown in Fig.3.11 and Fig.3.12, respectively. Since there

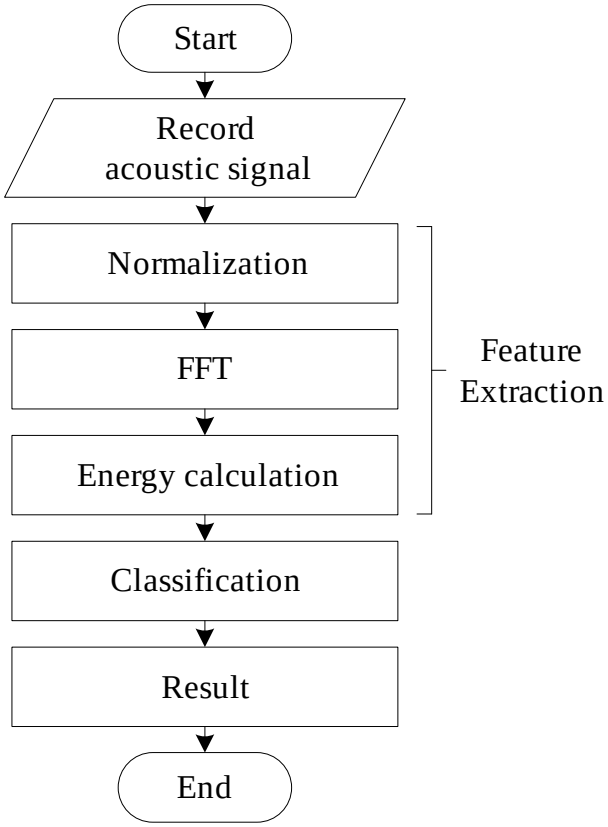


Figure 3.7: Procedure of acoustic-based fault diagnosis for legged robots (AFL).

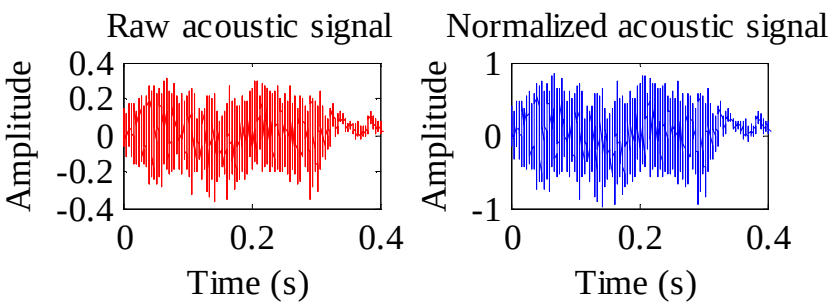


Figure 3.8: Raw signal and normalized signal of healthy joint.

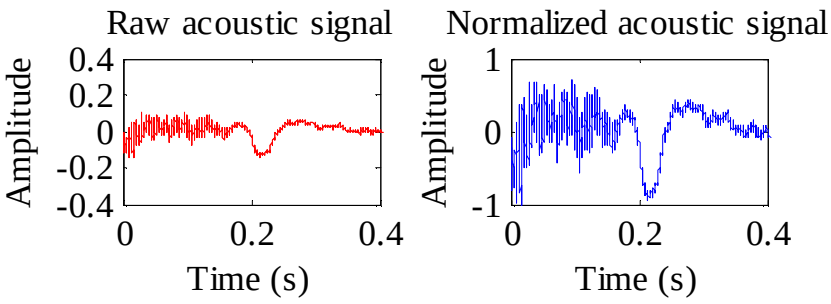


Figure 3.9: Raw signal and normalized signal of unhealthy joint.

Table 3.1: The f_L and f_H corresponding to frequency band.

Serial number	f_L-f_H (kHz)	Serial number	f_L-f_H (kHz)
1	0.1-2.1	5	8.1-10.1
2	2.1-4.1	6	10.1-12.1
3	4.1-6.1	7	12.1-14.1
4	6.1-8.1	8	14.1-16.1

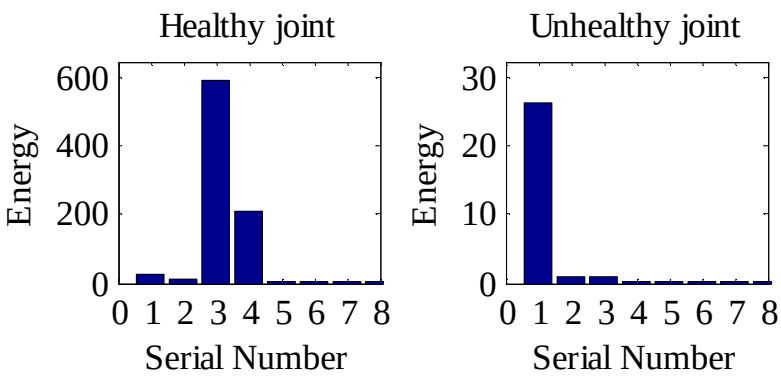


Figure 3.10: Example energy of each frequency bands calculation of healthy and unhealthy joint sounds.

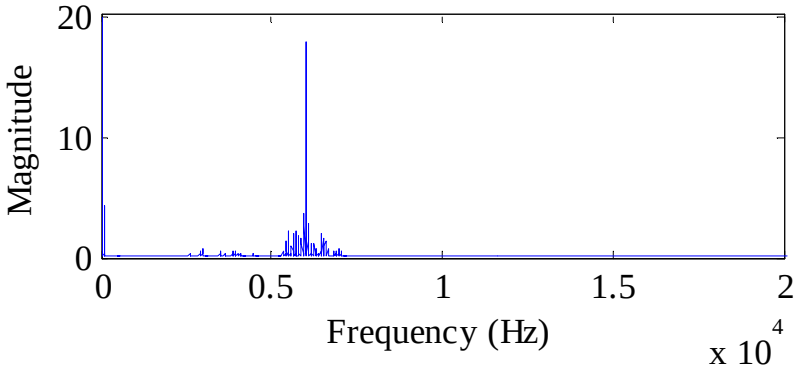


Figure 3.11: The frequency spectrum of healthy joint acoustic signal.

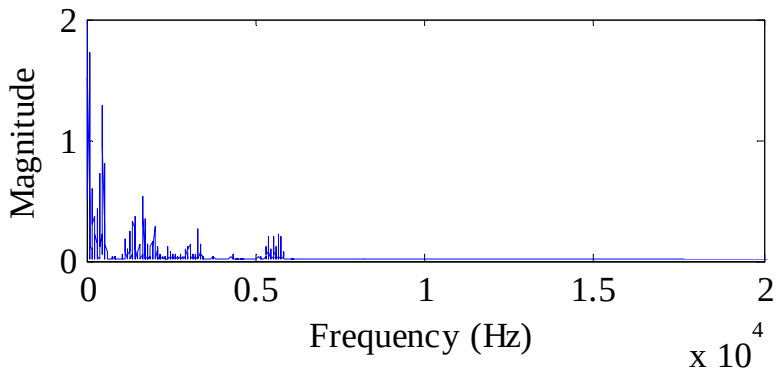


Figure 3.12: The frequency spectrum of unhealthy joint acoustic signal.

are so many frequency spectrum provided by FFT, meaning that some data are redundant, these data will be processed in the next step to minimize the number of features.

Energy calculation:

The energy of each frequency band is manipulated as a feature for classification process which can be calculated as follows:

$$Energy = \sum_{k \in S} |A(k)|^2 , \tag{3.12}$$

where

$$S = \{k : f_L \leq f(k) < f_H\}, \tag{3.13}$$

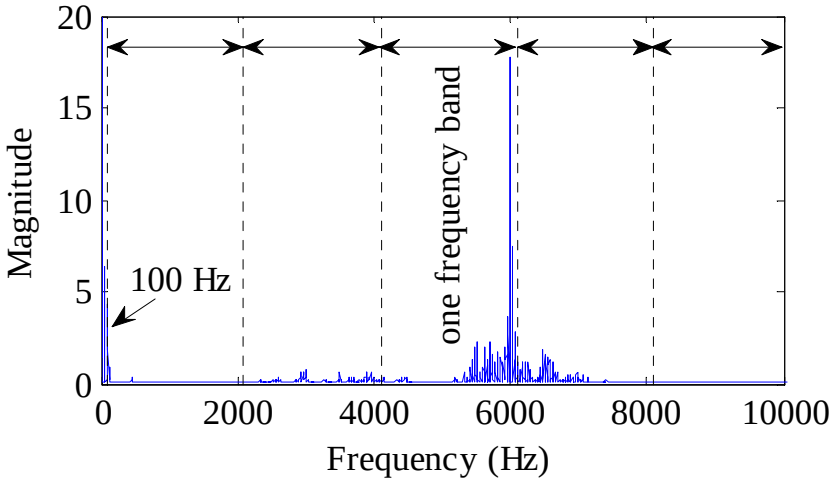


Figure 3.13: Energy calculation of each frequency band, set as 2000 Hz.

f_L and f_H are low and high frequencies of the frequency bandwidth, respectively. In this study, the frequency bandwidth is set as 2000 Hz. According to the experiments, there are various kinds of system noises occurred while operating. In order to make this system become more precise, the values allocating under 100 Hz are not included in this operation as shown in Fig.3.13, meaning that the parameters f_L and f_H of the 1st band are set as 100 and 2100 Hz, severally. The bands corresponding to frequency range can be seen in Table 3.1, and the examples of feature extraction of healthy and unhealthy joints are illustrated in Fig.3.10.

3.3.5 Classification

The AFL method employs the concept of fuzzy logic to classify the status of the joints of legged robot. In recent year, the fuzzy logic is used to detect the fault of induction motor in [6] and to monitor the status of vehicle online in [40], successfully. The advantage of the fuzzy logic classifier (FLC) is that it is more practicable than the conventional learning-based approaches (e.g., ANN) which is not only delicate to analyze but also the slow speed of training process [1]. However, the FLC can be operated with a limited number of input variables due to the fact that the input space will be divided into fuzzy regions. For AFL method, the energies of first-five frequency bands are selected as a feature vector.

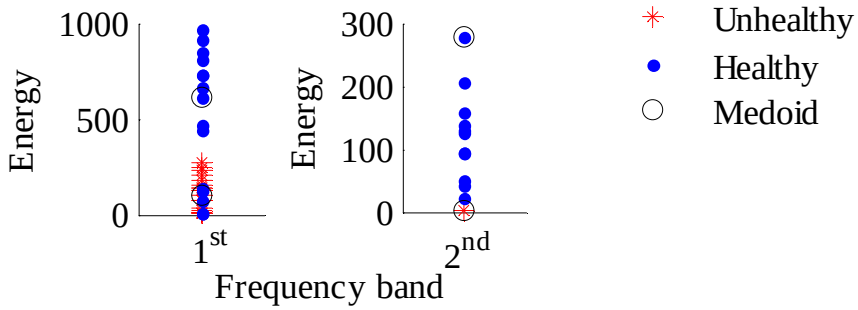


Figure 3.14: The labeled data of the 1st and 2nd frequency bands, preprocessed experimentally as database of classification.

As shown in Fig. 3.10, there is a significant disparity between the healthy and unhealthy values allocated in the first-five frequency bands. There is two fuzzy sets, *small* and *large*, defined for each feature. According to the pre-collected database, in certain features, it is improper to separate the region efficiently due to the overlapping region, as illustrated in Fig. 3.14. In such case, the small and large sets are predefined by concerning of the medoid points of each labeled feature, guaranteeing that most of data are still classified correctly.

In this study, there are four methods developed to classify the abnormality of joints based on the fuzzy classification.

Fuzzy Rule (FR):

The method is functioned along with the general concept of fuzzy logic by isolating the input space into fuzzy regions. Figure 3.15 shows the input space of two frequency bands, i.e., the 1st and 2nd frequency bands, with 20 data points of healthy joints and 20 data points of unhealthy joints. It is noted that there are more points of healthy joints allocated out of range illustrated in the figure. For this aspect, the fuzzy rule is simply defined, as shown in Table 3.2, in which x_1 is a member of the 1st frequency band and x_2 is a member of the 2nd frequency band. As stated by the fuzzy rule, it is noticeable that two classes can be classified efficiently with two sets of features. However, it will become more complicated to classify the data when the number of features is higher than two in consonance with Fig.3.16.

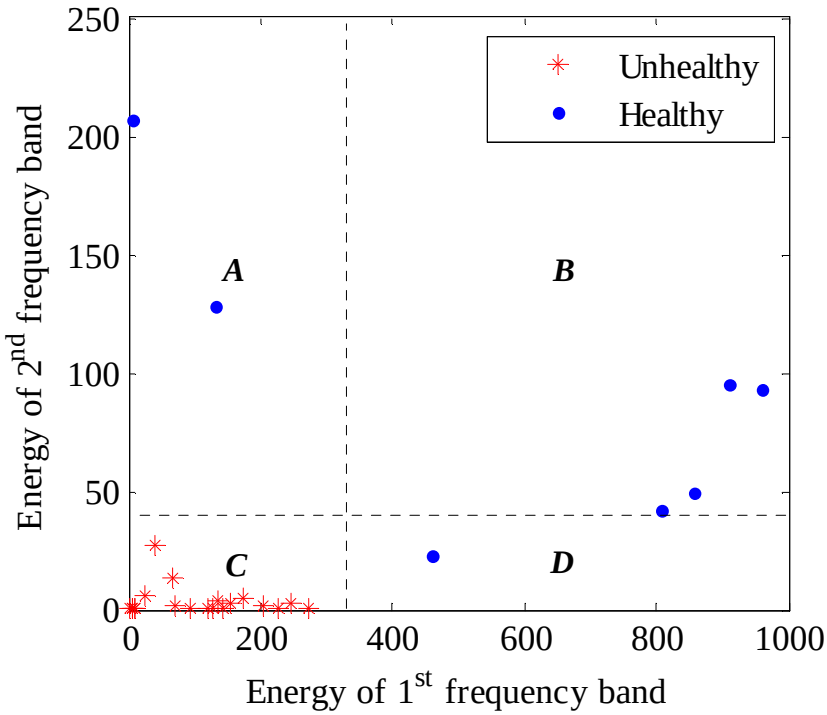


Figure 3.15: The input space of data from two frequency bands.

Table 3.2: Fuzzy rule for classification of healthy and unhealthy joints of legged robots.

Fuzzy Rule	Area
if x_1 is *small* and x_2 is *large* then Class 1	***A***
if x_1 is *large* and x_2 is *large* then Class 1	***B***
if x_1 is *small* and x_2 is *small* then Class 2	***C***
if x_1 is *large* and x_2 is *small* then Class 1	***D***

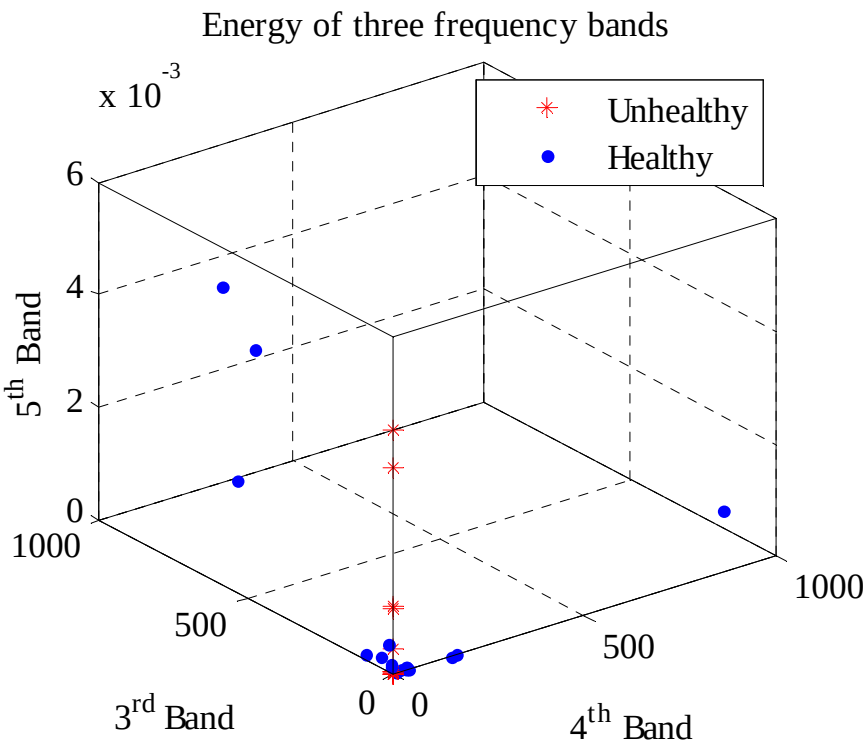


Figure 3.16: The input space of data from three frequency bands.

Mean of Fuzzy Set (MF):

To avoid the complexity of determining fuzzy region in an input space, the MF method uses the fuzzy values – the degrees of truth and probabilities range between 0 and 1 – to identify whether healthy or unhealthy by calculation the mean value of five features. Since there are two fuzzy sets for each feature, the *small* and *large* sets are directly defined as unhealthy and healthy sets, consecutively. The result of classification is governed by comparing the mean values of probabilities between two sets, as Eq.(3.14),

$$output = \begin{cases} \frac{prob_h+1}{2}, & \text{if } prob_h > prob_u \\ \frac{1-prob_u}{2}, & \text{if } prob_h < prob_u \\ -1, & \text{otherwise} \end{cases}, \tag{3.14}$$

where $prob_h$ and $prob_u$ are set as the mean values of probabilities of healthy and unhealthy, sequentially.

Medoid of Fuzzy Set (MDF):

Medoid is a value of elements in the set which represents the center of clustering. This method is similar to the MF method, but the medoid value is used instead of mean value. Additionally, the final output is also calculated using Eq.(3.14).

Voting of Fuzzy Set (VF):

The voting method is adapted to used as a decision making process in this approach. It is noted that the values of probability that is higher than 0.75 are also counted in voting process. For example, the result of voting for set $A = \{0.8, 1, 0.5, 0, 0.9\}$ is 1.

3.4 Results and Discussion

3.4.1 Simulation Results of PLI Algorithm

The performance of proposed algorithm (PLI) is certified by numerical simulation, and compared with the GA-based method as it uses only internal sensor, likewise [103]. There are three of broken legs validated in the experiments, which are one leg-loss by half, one leg loss and two leg-loss, illustrated in Fig. 3.17. Two

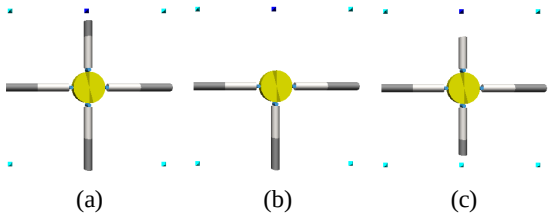


Figure 3.17: The damaged robot tested in the experiment: (a) first lower limb-loss by half, (b) first leg-loss and (c) first lower limb-loss and third lower limb-loss by half.

algorithm, PLI and GA-based method, are programmed to perform with the same sets of parameter as population of 80 particles and 25 iterations with 5 ms sampling rate. Each algorithm will perform 10 times, and the average of root mean square (RMS) error will be measured. The RMS error can be calculated as follows:

$$RMS_{error} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{3.15}$$

where $i = 1, 2, ..., n$ and e is the error of lengths between the actual damaged robot and the new updated model obtained from the simulation. The performance of PLI is also measured by varying the number of particles in the population, and this experiment collects the number of iteration that the best models are discovered. In addition, the noise and sampling rate tests are conducted to evaluate the performance of proposed method.

One-leg loss by half (Case A)

In this test case, two algorithm are conducted to verify the broken robot with lower limb-loss by half, and the average RMS errors are measure. The example of result is shown in Table 3.3. The PLI can provide more accurate results, compared to GA-based method, and the average RMS error is also lower as the RMS error of PLI and GA-based method are 0.094 and 9.157, subsequently. The main reason why GA-based method product more error is that it employ the switches mounted at the end of robot's legs. In case of damaged leg, the responses of the switch cannot be measured correctly.

One leg loss (Case B)

The PLI still performs well and provides good result in term of error, as illustrated in Table 3.3. Due to the same reason as aforementioned in 4.1, the GA-based method, further, provide big error while operating. Since this benchmark is simpler than the first one – as no shorten leg to make an impact movement, the error of GA-based method is less than the result in Case A, as 16.408. Still, the proposed can make good result as RMS error is 0.147.

Two leg loss (Case C)

This is the most difficult test compared to others because there are two broken legs, one leg-loss by half and one lower leg-loss by half. The error of PLI algorithm is still lower than GA-based method because of the same reason. However, the GA-based method can provide feasible result as illustrated in Table 3.3. The RMS errors of PLI and GA-based Method are 0.196 and 3.133 by order.

Number of particles

To ensure that PLI algorithm can discover the new model correctly, the number of particles is varied to test the stability. The result of this test can be seen in Fig. 3.18. The number of particles is set at 40, 60 and 80 particles. The experiments are conducted for 30 times per each case, and the average numbers of iteration are shown in the graph. In Case A, the result shows that increasing number of particle leads the robot to find the new model rapidly. However, in Case B, a low number of particles can perform faster because this situation is simpler than other so that more number of possible solution can affect the system failure. In the most complex experiment, Case C, the results are therefore same as Case A since it is difficult to detect the correct model, a large number of feasible solutions will make the robot to get the correct model faster. Therefore, the number of particles can affect the performance of leg-loss identification in term of time spent. On the other hands, in non-complicated situation, a low number of particles can be employed to achieve better solution.

Noise Analysis

Since the PLI algorithm is experimented only in simulation, the noise effect is considered as well. This experiment is done with Case A with 5 ms sampling

Table 3.3: The length of robot's legs (in cm) from PLI and GA-based Method.

Leg Number		Upper Limbs				Lower Limbs				RMS Error
		1st	2nd	3rd	4th	1st	2nd	3rd	4th	
Case A	*Actual*	40	40	40	40	20	40	40	40	-
	PLI	40	40	40	40	19.745	39.991	39.991	39.921	0.095
	GA	33.647	40	32.941	40	0	33.647	48.941	32.235	9.157
Case B	*Actual*	0	40	40	40	0	40	40	40	-
	PLI	0	40	40	40	0.4152	40	40	40	0.147
	GA	33.647	40	29.176	40	27.058	40	29.412	32.235	16.408
Case C	*Actual*	40	40	40	40	0	40	20	40	-
	PLI	39.72	40	40	40	0.079	39.609	19.739	39.94	0.196
	GA	40	41.882	32.941	40	1.647	40	24.235	37.882	3.133

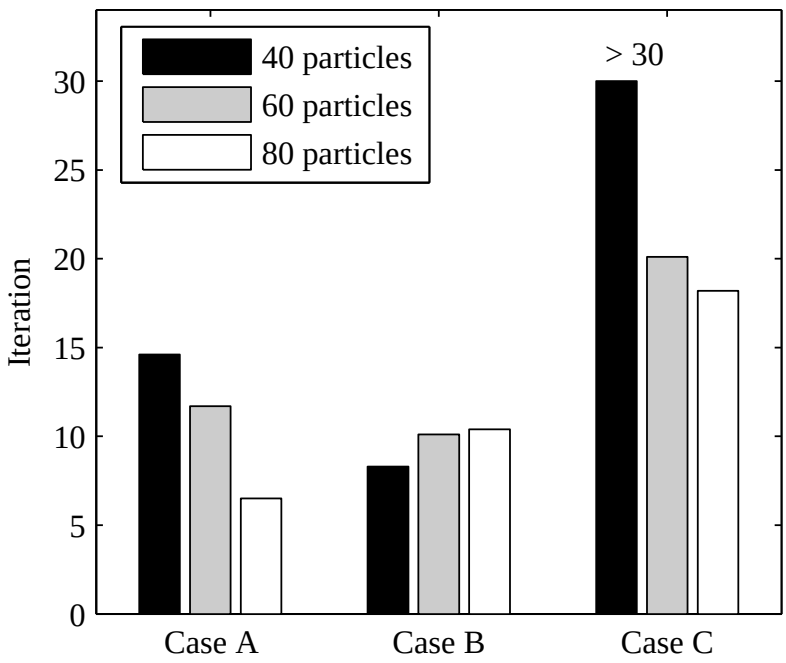


Figure 3.18: Results of number of particles test (average iteration that robot discovered the best solution).

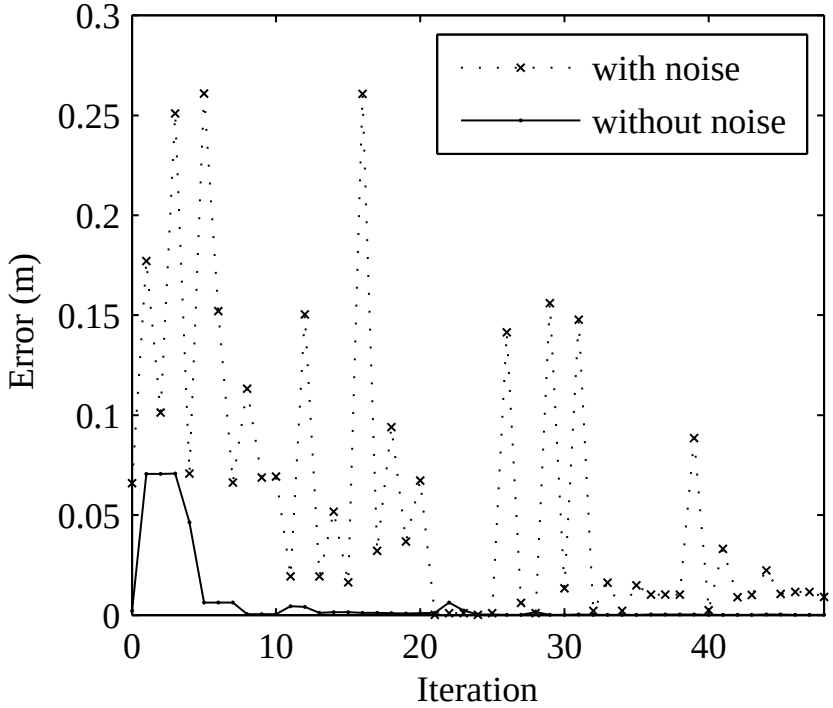


Figure 3.19: Results of error with noise test.

rate and 80 particles, and the Gaussian noise is added into the signal to simulate the noise of the accelerometer and gyroscope which are used in the experiment. However, the noise in the real system is not similar as simulated, but the aim of this test is to ensure that the PLI can perform with the noisy condition. The result shows that, with noise, the error of the system is bigger than that without noise situation; however, the error at the end of the simulation becomes smaller (acceptable result with 2 cm error) as shown in Fig. 3.19. It means than the PLI algorithm is feasible to apply in noisy systems. In additional, the PLI will be demonstrated with an actual damaged robot in the next section.

Sampling rate

The sampling rate of the signals are also tested in the experiment, and it has been evaluated with case A, without noise and 80 particles. As shown in Table 3.4, the result shows that varied sampling rates, 5, 10, 20 and 100 ms, do not cause

the major errors. Although the large sampling rates have not been tested in the experiments, in general robotics system, the 100 ms of sampling rate is possible for robot to operate properly.

Table 3.4: The length of robot's legs (in cm) of sampling rate test.

Sampling Rate		5 ms	10 ms	20 ms	100 ms
1st leg	Upper Limb	40	40	40	40
	Lower Limb	20.009	19.999	20.017	20.021
2nd leg	Upper Limb	40	40	40	40
	Lower Limb	39.972	39.999	40	40
3rd leg	Upper Limb	40	40	40	40
	Lower Limb	39.982	39.879	40	39.999
4th leg	Upper Limb	40	40	40	40
	Lower Limb	39.98	39.999	40	40
RMS error		0.019	0.181	0.003	0.005

3.4.2 Experimental Results of PLI Algorithm

To be certain that the proposed method is feasible to employ with practical robots, two experiments are conducted in this study. The PLI algorithm is programmed and tested with 2-DOFs-quadruped robot with external Inertial Measurement Unit (IMU) – which is used to measure the orientation of robot's body. Two cases of broken legs, one leg ripped out and three limbs ripped out, are conducted in this test.

Experimental Setup

In the experiment, a robot is made up with four legs consisting of two degrees of freedom per each leg. Eight Dynamixel servo motors are employed as actuators for each joint, illustrated in Fig. 3.4. The size of robot's body is set as a square with 16 cm width (l_b). The lengths of lower limb (l_l) and upper limb (l_u) are 6 cm and 8 cm, respectively. The quadruped robot are controlled through the computer

with FDIII-HC board and USB cable. The orientation of robot is measured via x-IMU, high-performance and well-calibrated IMU, which is connected to computer wirelessly with Bluetooth technology.

For simulation side, the parameters, w, c_1 and c_2, are set as 0.99, 0.5 and 0.5, in order. The weight balance is set as 0.99 since it is expected to be minimized over time – additionally, the solutions of evolutionary algorithm (like PSO) will be getting optimized as time passes. The values of parameters, c_1 and c_2, are identical because of leading the particles to follow both global and local solutions. Even the proposed algorithm can operate successfully in the simulation, there is a minor contrast between simulation and real-world environment. To avoid such problem, the high-performance sensor is selected to obtain the orientation's values in this study; however, there is still a gap between simulation and a practical robot in the experiments as shown in Fig. 3.20. The values of orientation achieved by x-IMU are rather diverse from the values calculated in the simulation. Even if the values are not completely similar, the tendency goes somewhat in the same direction. The proposed algorithm, PLI, is modified to overcome the problems of dissimilar orientation. Firstly, the velocity (v), used to update the particle, is changed from Eq. (3.16) as follows:

$$v_{id} = (1 - cw) \cdot [w \cdot v_{id} + c_1 \cdot \text{rand}() \cdot (p_{id} - x_{id}) + c_2 \cdot \text{rand}() \cdot (p_{gi} - x_{id})] \quad (3.16)$$

Since there is a difference between simulation and actual robot – making this term $(1 - cw)$ become 0 gradually, transferring Eqs. (3.4)-(3.6) leads the velocity to become convergent rapidly. The position (x) is updated by using PSO original equation Eq. (3.5). Secondly, the probability of mutation ($prob$) is adjusted to 0.50 to avoid particles to allocate in local optimal. Thirdly, the selection of new updated model is done by the current and previous fitness values (f_c and f_p). The final solution will be achieved if f_c and f_p are greater than the desired value, set as 0.95 in this study. In additional, the sampling rate, to read the sensor values, is defined as 20 ms, and the candidate models in the search space are programmed to perform for 250 ms. The population of particles is set as 80 particles.

One leg ripped out (Case D)

According to the result written in Table 3.5, the PLI algorithm provide the acceptable result with 3.392 RMS error. The finding process is done with 6 iterations, and the maximum fitness value is 0.953. A main obstacle which causes an error can be noted as the difference between the orientations of real damaged robot and

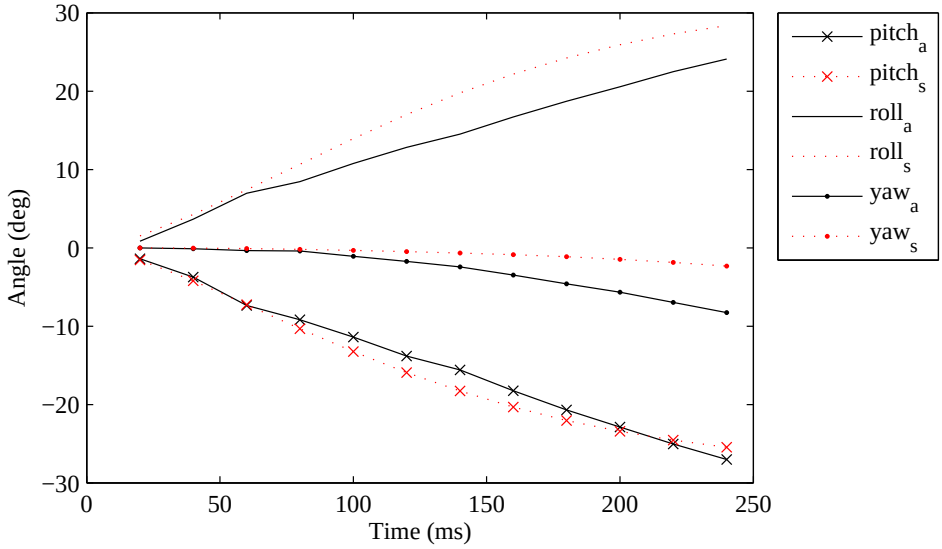


Figure 3.20: The orientation of robot's body between actual robot (a) and simulation (s).

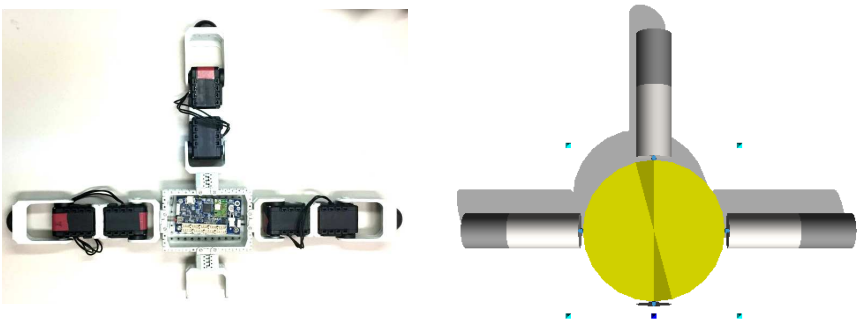
candidate models in the simulation. Therefore, it can be concluded that the proposed method, PLI, can assist the broken robot to discover a new updated model. Figure 3.21 shows the actual robot and the new model found by PLI algorithm.

One leg and one limb ripped out (Case E)

This test case is similar to Case C in previous section, and it is hard to comprehend the final solution. The PLI cannot discover the new model efficiently as shown in Table 3.5. Due to the same reason causing to Case C, the RMS error is quite high as 21.133. The main problem providing the error is not only the difficulty but also the gap between simulation and real world; nevertheless, the PLI algorithm can give a rough structure of new model in this experiment as illustrated in Fig. 3.22. It is worth noting that if the gap between simulation and actual robot can be minimized, the successful rate can be increased, following the simulation of Case C.

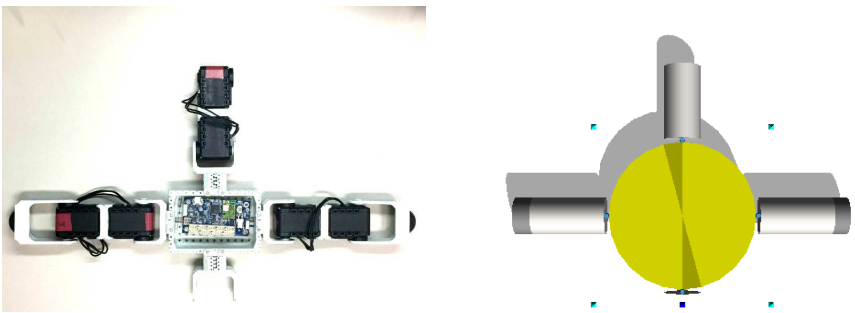
3.4.3 Results of AFL Algorithm

In the experiment, there are four cases conducted to evaluate the robustness of the classification. The output of classification process will be allocated between



Actual Robot Simulation

Figure 3.21: The actual damaged robot and the updated model provided by PLI of Case D.



Actual Robot Simulation

Figure 3.22: The actual damaged robot and the updated model provided by PLI of Case E.

Table 3.5: The length of robot's legs (in mm) from PLI in the experiments.

	Leg Number	Upper Limbs				Lower Limbs				RMS Error
		1st	2nd	3rd	4th	1st	2nd	3rd	4th	
Case D	*Actual*	0	80	80	80	0	60	60	60	-
	PLI	0	80	80	80	0	61.132	61.609	50.61	3.392
Case E	*Actual*	0	80	80	80	0	60	0	60	-
	PLI	0	80	80	80	0	16.6	0	18.9	21.133

Table 3.6: Results of classification in healthy robot case.

	Joint Number							
	1	2	3	4	5	6	7	8
Actual	◯	◯	◯	◯	◯	◯	◯	◯
ANN	◯	◯	◯	◯	◯	◯	◯	◯
FR	◯	◯	◯	◯	◯	◯	◯	◯
MF	◯	◯	◯	◯	◯	◯	◯	◯
MDF	◯	◯	◯	◯	◯	◯	◯	◯
VF	◯	◯	◯	◯	◯	◯	◯	◯

◯ = Healthy, × = Unhealthy

0 and 1, referring to healthy and unhealthy class. If the output is close to 0, it means that the joint will be arranged in the healthy class. On the contrary, the joint will be sorted into the unhealthy class if the output gets close to 1. The performance of each method will be appraised by calculating Mean Square Error (MSE), and the accuracy of each method will be determined as well.

Healthy robot

According to Table 3.6, the results show that all algorithms can classify the class efficiently. However, there are some minor errors – according to the numerical results – with ANN and MF methods as 0.0008 and 0.005, respectively. The error occurring with ANN's output is possibly caused by the improper training process; yet, the result is acceptable. The overlapping region of classes slightly leads the insignificant mistake in MF method. Conversely, the FR, MDF and VF can avoid the problem of overlapping region successfully.

One leg locked

In this experiments, all algorithm can operate properly except ANN with the similar problem as seen in Table 3.7. The ANN network can become overfitting if the traninng process is not treated in a good manner. While testing the training set with this ANN network, it can classify the data into the correct classes successfully in the experiments. Thus, it can be concluded that the error occurred because of overfitting network. The MF produces less error than ANN, and the results are therefore acceptable. Again, the FR, MDF and VF still provide the

Table 3.7: Results of classification in one leg locked case.

	Joint Number							
	1	2	3	4	5	6	7	8
Actual	◯	◯	×	×	◯	◯	◯	◯
ANN	◯	◯	×	×	◯	◯	◯	×
FR	◯	◯	×	×	◯	◯	◯	◯
MF	◯	◯	×	×	◯	◯	◯	◯
MDF	◯	◯	×	×	◯	◯	◯	◯
VF	◯	◯	×	×	◯	◯	◯	◯

◯ = Healthy, × = Unhealthy

Table 3.8: Results of classification in one leg non-powered case.

	Joint Number							
	1	2	3	4	5	6	7	8
Actual	◯	◯	◯	◯	◯	◯	×	×
ANN	◯	◯	◯	◯	◯	×	×	×
FR	◯	◯	◯	◯	◯	◯	×	×
MF	◯	◯	◯	◯	◯	◯	×	×
MDF	◯	◯	◯	◯	◯	◯	×	×
VF	◯	◯	◯	◯	◯	◯	×	×

◯ = Healthy, × = Unhealthy

good results in terms of errors.

One leg non-powered

Similar to previous two experiments, only ANN and MF method generate the errors as 0.125 and 5.875×10^{-5}, respectively, while FR, MDF and VF are still functioning sufficiently with 0.000 error. Table 3.8 shows the experimental results of this test case.

The 40 acoustic signals of healthy and unhealthy joints

This experiment is conducted to measure the accuracy of the proposed methods. Forty random acoustic signal of healthy and unhealthy joints (20 signals for each

		Actual	
		Healthy	Unhealthy
Predicited	Healthy	TP 18	FP 2
	Unhealthy	FN 0	TN 20

(a)

		Actual	
		Healthy	Unhealthy
Predicited	Healthy	TP 20	FP 0
	Unhealthy	FN 1	TN 19

(b)

		Actual	
		Healthy	Unhealthy
Predicited	Healthy	TP 20	FP 0
	Unhealthy	FN 0	TN 20

(c)

Figure 3.23: The confusion matrix of classification: (a) ANN, (b) FR and (c) MF, MDF and VF.

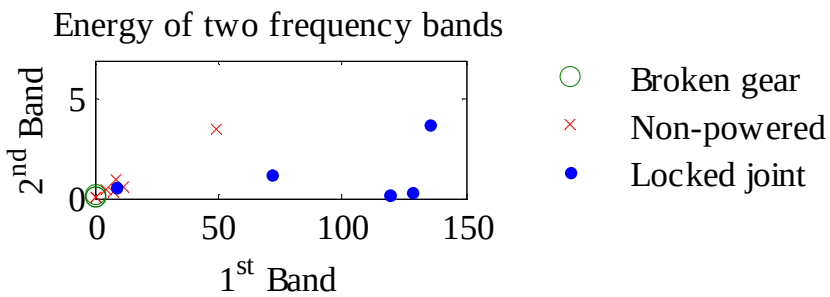


Figure 3.24: The data of three different types of faults.

class), are set to extract the feature and feed to the input of classifier. Figure 3.23 illustrates the confusion matrix of all methods tested in the benchmark. It is noticed that MF, MDF and VF have the equivalent confusion matrix so that they are belonged to the same matrix. In addition, the variables, TP, FP, FN and TN, are referred to True Positive, False Positive, False Negative and True Negative, serially. The accuracy of the classification can be determine as

$$Accuracy = (TP + TN)/Total\ population. \tag{3.17}$$

The ANN provides poor results compare to others in terms of accuracy as 0.95 while FR can classify the faults better than ANN with 0.975. On the other hands, MD, MDF and VF can perform efficiently with 100 percent accurate.

Types of faults classification

The types of faults are detected in this experiment. There are three types of fault tested that are broken gears, non-powered and locked joints. The ANN and FR methods are employed to cluster the types of faults. According to the Fig. 3.24, it is complicated to classify the types of faults since some data are extremely overlapped. In this case, the ANN with with 5 hidden layers of 5 nodes cannot be trained successfully so that the 10 hidden layers are constructed to perform in place. For FR method, the three basic fuzzy rules are utilized to overcome this problem. The results show that FR method can classify the types of faults more accurate than ANN method with 5% detection error. Although the network was trained properly, the ANN method cannot perform well with 40% detection error. Additionally, the error occurred in this experiment are caused by the fact that some data are quite similar.

3.5 Summary

This chapter has proposed to develop the leg-loss identification algorithm based on particle swarm optimization method and normalized cross-correlation algorithm called PSO-based Leg-loss Identification (PLI). The procedure of detecting the broken legs is done by comparing the actually damaged robot and a number of candidate robots in the search space with the orientation signal. The responses of actual damaged robot and candidate model are compared by the cross-correlation method, which is also utilized as the fitness function in PSO. The PSO algorithm is employed to evolve the candidate models to provide the same behavior as the damaged robot. In the experiment, the results of PLI are compared with GA-based method, and they show that PLI is better in term of efficiency. In addition, the number of particles is varied to evaluate the performance of the proposed algorithm, and it can be summarized that, in complex situation, a large number of particle can assist the robot to detect a failure part faster, but in case of simple problem, a limited number of particle can provide more robustness. However, in the future, it is planned to test the performance of the algorithm in various problems. To be certain that the PLI can handle the noisy system (represented the practical situation), the noise analysis is also conducted, and the result shows that even if noisy signals provide more error than non-noisy signals, the error becomes small at the end of the process. The changing of sampling rate has experimented

as well. The error between the best candidate model and the actually damaged robot is very small even the sampling rate increased ten times. Moreover, the experiments also show that the PLI can discover the new updated model with the actually damaged robot. However, the difference between simulation and practical situation makes an error to the final model but it is acceptable as the structure of the model is similar to the damaged robot. It can be concluded that the proposed method (PLI) is feasible to detect the broken leg by discovering the new model, which will be beneficial for the legged robot to create the alternative walking behavior after damaged. Additionally, the performance of the PLI can be improved by adding the transferring module between simulation and the real world, which decreases the gap and errors.

Furthermore, we have found a novel method of fault diagnosis of a joint by means of acoustic processing. The process has been done by extracting the features of sounds with the FFT and the energy of the frequency band. The classification process was operated based on the concept of fuzzy logic. Four different classifications are conducted in this paper. According to the experiments, it can be concluded that the proposed methods, FR, MDF, and VF, can detect the fault of joint efficiently. Due to the simplicity, the VF method is the best manner to diagnose the faults. The MDF method is also operated successfully without error, but it is costly to find the medoid. Even MDF can provide the same level of accuracy, some minor errors occur with output. Moreover, the FR method is feasible to employ with a non-complex and small set of a cluster which is more simple than the conventional learning system. In order to prevent from the overlapping region in input space, it is planned to apply the fuzzy classification by weight which employs optimization algorithms to avoid the bad interpretability in the future. Moreover, the additional sensor will be integrated to increase performance and accuracy.

Chapter 4
Self-recovery Methods

4.1 Introduction

In this chapter, the novel self-recovery algorithm will be explained deeply in detail. Although the legged robots are more flexible than wheel-based robots, the one drawback is that they are difficult to operate when getting damaged. Researchers have discovered and developed the solutions to overcome this problem. The ability of system that can continue functioning in presence of faults is known as fault tolerance [22]. In fault tolerant system, there are four processes to be considered which are fault detection, fault location, fault containment, and fault recovery. Fault detection is a process to determine that a fault has occurred. Fault location is a process to determine where a fault has occurred. Fault containment is a process to isolate a fault from the system. Fault recovery is a process to make the system get recovered from fault. Fault-tolerant gait planning is the recovery method used with multi-legged robots after a failure has occurred and hampers its ability to walk or maintain stability [100]. Accordingly, the gait planning techniques were proposed to overcome failures, such as joint lock and fault actuators, in [13, 53, 99]. Also, the analysis of fault tolerant gait and motion planning were presented in [19, 69]. Faults were recovered for modular robots reported in [89]. The modular robots did not resemble themselves but plan the new motion with CPG. Moreover,

According to the researches published in the past, we can categorize the recovery methods for damaged legged robots mainly into four groups which are evolutionary-assist method, gait transition method, task-based method [57] and learning method. The remarkable research was published in 2006 by Josh Bongard's group. They proposed the algorithms to let damaged robot be able to walk again [103]. The robot will start the process by identifying its curent model and employing the evolutionary algorithm to find the behavior (self-learning method)

that provides the best movement. They showed the impressive results that the robot with the broken legs can travel forward surprisingly. However, the main downside of this method is that the robot cannot perform successfully all the time. The similar methods are also published in [51, 63]. They employ the evolutionary methods, such as GA, to crate the sequential actions that can control robot to move forward. In [42], the GA was also employed to perform gesture reconfiguration of the humanoid robot when the joint failure occurred. However, these kind of method is a time consuming task. The robots have to execute actions again and again until they receive the maximum outcome. The problems are not only the time spent but also the lack between real robot and simulation. Typically, the evolutionary method need the numerical processing that is tough to operate within the robot itself. Therefore, the researchers utilized the simulation model on the the computer to discover the maximum-distance-traveled actions so that the mismatch between the simulation and reality occurs. In [49], they employ both simulation and reality process. Firstly, the robot will find the good action in the simulation, and then operate the action with real robot. This method provide the very result in terms of time consuming and moving ability. Since it still spends some time to process, the same research group came up with the new amazing idea which was published in [15], called as "the robot that can adapt like animal". The results of this method is outstanding. They use the trial-and-error technique combined with the large search space. The robot perform a number of possible action in the simulation beforehand (approximately one week for this process) and they will be store in the search space to be pick up when the robot need to adapt. Using this algorithm, the robot spent less than two minutes to get recovered. Even though this method can apply successfully with hexapod robot and robot arm application, the implementation with only 4 leg robot was not reported. In [65], the gait transition with central pattern generator or CPG was proposed. The change of frequency is applied to CPG in order to make robot overcome the damage. This method use the multiple chaotic CPGs with online-learning to let robot perform the fault tolerant movement. The result show that the robot can move the the desire path and reach the goal. However, it employs a number of infrared sensors and force sensors to be feedback sensory signal to the system. As a result, these can deal with some specific situations as reported in [65].

Due to the fact that not much of existing method have dealt with the legged robot that has the number of legs less than 4. Only the works published in

[51, 63], the algorithms were tested in the scenario that the robot have only 3 legs. Unfortunately, these methods suffer from a number of pitfalls as mentioned earlier. In this study, we also uses evolutionary method to find the action of the damaged robot. The same concept from [63] was conducted with the change of evolutionary algorithm, which the PSO was applied instead of GA. Since it require a number of simulation and unsatisfying results, the new method of control method based on the task-predefined and learning method has been developed. The main inspiration is from the living creature in the nature. Mudskipper is a fish that can do crawling on the mud with only two fins. We define the structure of mudskipper as 2-DOFs and create the action loop with sine and cosine function. Additionally, the Q-learning method is integrated with the mudskipper-inspired behavior to increase the performance of the robot to move faster and to be more efficient. The proposed algorithm is called as the self-learning mudskipper-inspired crawling method (SLMIC) [11].

4.2 Self-recovery Method based on Evolutionary Algorithm

We adopted a modified version of the work described in [51, 63]. PSO was employed over GA due to its simplistic modeling and optimization convergence. The method can be described into three main parts, namely, movement sequence coding, objective function, and evolutionary process.

4.2.1 Movement Sequence Coding

As aforementioned, we employed smart electric actuators, Dynamixel^{TM}, to control the joints to the programmed angles. The PSO particles were designed based on the idea of sequence actions. Each motor is required to perform five sequential actions. The robot performs each action for t seconds ($t = 0.2$). Every joint is driven by a self-desired action at the same time. Hence, we can define the actions of each motor as follows:

$$S_i = [Ang_{i1}, Ang_{i2}, ..., Ang_{i5}] \tag{4.1}$$

where S is the sequence of actions, i the number id of joint motor, and Ang the control motor angle. Then, the PSO particles (P) can be described using Eq.(4.2).

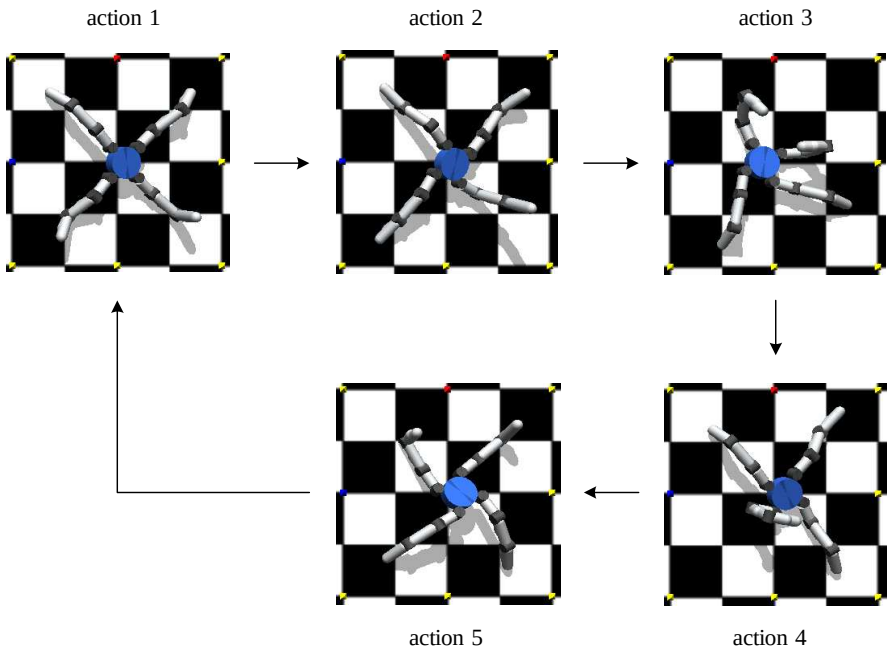


Figure 4.1: Example of robot movement with sequential actions. Robot will start executing from action 1 to action 5 continuously in round-robin fashion.

$$P = [S_1, S_2, ..., S_N] \tag{4.2}$$

where N is the number of joints of the legged robot. The proposed robot has 12-DOFs so N is equal to 12. The overall parameters of one particle are $5 \times 12 = 60$ values. An example of robot movement is shown in Fig. 4.1. The robot performs 5 actions iteratively until it completes the desired task.

4.2.2 Objective Function

The criteria to achieve the recovery movement of a damaged robot can be set in a form of an objective function. This function is used to judge which solutions can provide the best result. To overcome damage, movability is the main objective of the recovery process. If the robot cannot move properly, it will be impossible to fix the robot. By contrast, if the robot can move to the desired position, we can repair the broken parts of the robot and can send it back to complete its mission. Accordingly, we set movability as the travel distance in a given direction. The damaged robot that can move faster to the required position is said to have the

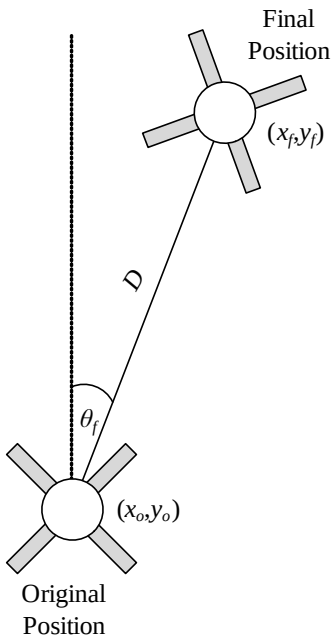


Figure 4.2: Trajectory of damaged robot traveling from the original position to final position.

maximum of movability. The distance traveled by the robot can be calculated after the robot moves for T seconds. Figure 4.2 shows the performance of the broken robot traveling from the original position to the final position. The distance traveled D can be determined using a basic geometrical approach as follows:

$$D = \sqrt{(x_f - x_o)^2 + (y_f - y_o)^2} \tag{4.3}$$

where x_o and x_f are the position of the robot along the x-axis of the original and the final position, respectively. In addition, y_o and y_f are the robot position along the y-axis of the original and the final positions, respectively. To follow the required direction, the parameter θ_f is put mathematically in the objective function F to decrease its value in case if the robot travels the wrong distance. In addition, we focus on straight line motion in this study. If θ_f becomes 0, it means the robot has traveled straight and F is maximized.

$$F = e^{-\theta_f^2} \cdot D \tag{4.4}$$

where θ_f denotes the different angles between the original and the final positions. Moreover, the calculated value of F is used to process the evolutionary method.

4.2.3 Evolutionary Process

The evolutionary process is similar to the PSO process, as discussed in the previous sections. A flowchart of the overall process is shown in Fig. 4.3. The process starts with the generation of random populations of n particles (P). Thereafter, all particles execute the actions for time T. Next, the performance of each particle is judged based on the value of the objective function F. The evolutionary method is execute under the following conditions:

- Normal: If a particle is allocated to this state, the action parameter will be updated according to the PSO rule.
- Capsizing: In practice, the robot attitude should be considered because sensors and loads, e.g. camera, are generally integrated on top of the robot. If the robot flips over during recovery, the sensor or loads may break. Therefore, the particles that cause the robot to flip over should reset all of their parameters randomly.
- Moving Backward: If the particles cause the robot to move backwards, their parameters should be set as random values, likewise.
- Mutant: Given the probability $prob < 0.2$, a few particles should be mutated to avoid the local maximum.

The process will be terminated if the objective function reaches the desired value or if the generation reaches the set value.

4.3 Mudskipper-inspired Moving Method

Due to the fact that quadruped robot need at least three legs to compensate the balance of its body [16], it is not feasible for the robot to execute the movement with standard balance after getting damaged. In the existing recovery methods, robots cannot walk properly even one leg has miner damage – also the method written in 4.2. As a result, we try to discover an alternative aspect to control the robot based on the specific actions. Since the action can be designed manually with careful observation, there will be a guarantee that the robot can keep itself in the good posture. To decide which actions to perform, we employ the natural inspiration from the fish called 'mudskipper', shown in Fig. 4.4. It is obviously

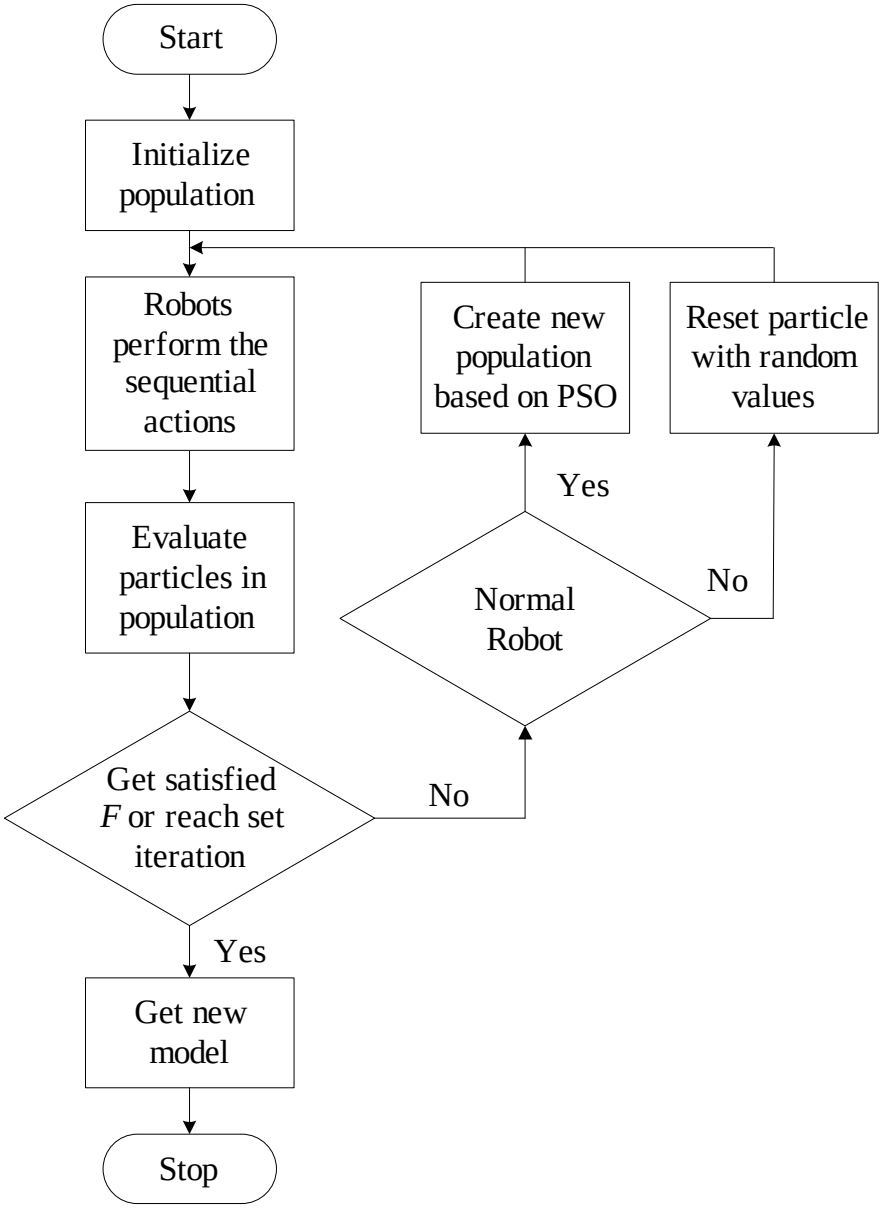


Figure 4.3: The procedure of evolutionary based self-recovery method.

Figure 4.4: Mudskipper laying on a rock.

seen that the adaptation of nature bring all living creature become suitable in which they live from million years ago to now. Mudskipper is one of survival species that adapt itself to be able to do not only crawling on land but also swimming in the water. We mimic the behavior of mudskippers crawling as it activates only two fins and one tail to create the excellent locomotion over the different types of terrains [91]. Recently, the mudskipper-inspired robots have been developed and analyzed. In [55], MuddyBot show the advantage of using tail and fins to improve the ability of movement. With the proves of this concept, it is feasible the this behavior can assist the movement of broken quadruped robot.

4.3.1 Mudskipper Behavior

In this study, we focus only on the operation of the two fins of a mudskipper. The simplified model of mudskippers fins is defined as a 2-DOFs system, containing two revolute joints with respect to the vertical and the horizontal directions. Figure 4.5(a) shows the design model of the mudskipper fins. According to the robot model discussed herein, a quadruped robot consists of 3-DOFs per each leg as shown in Fig. 4.5(b). Then, the angular rotations of θ_2 and θ_3 are set along reverse direction. Hence, we can describe the setting of each angle as follows: $\theta_1 = \phi_1$, $\theta_2 = \phi_2$ and $\theta_3 = -\phi_2$. To imitate fin movement, we performed numerical calculation using the sine and cosine functions of time t and one-time-moving

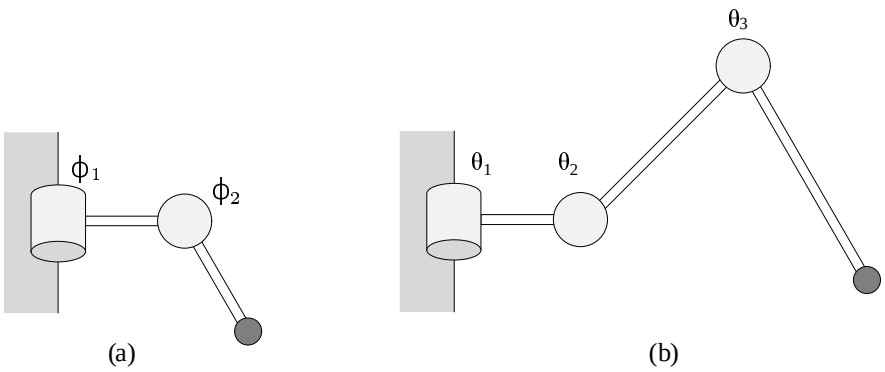


Figure 4.5: The model of mudskipper's fin and quadruped robot: (a) 2-DOFs mudskipper fin, (b) robot used in present study.

period T, as shown below:

$$\phi_1 = w_m \cdot \cos(\frac{2\pi t}{T}) \tag{4.5}$$

$$\phi_2 = h_m \cdot \sin(\frac{2\pi t}{T}) \tag{4.6}$$

where w_m and h_m are the width and the height of the moving trajectory, respectively. As can be seen from Fig. 4.6, the moving phase of the leg consists of two different phases, namely, swing and touch. At time $t = 0$ s, the leg will start from the beginning and cover a circular trajectory, and at $t = T$ s, the leg will end at the starting point. Note that the swing phase is the action in which the leg lifts off the ground, and the touch phase is the action in which the leg touches the ground.

To control the robot direction, the robot orientation θ_r is used as the control parameter to tune the direction of motion. A simple feedback control scheme is used in this case. The corresponding closed-loop control diagram is shown in Fig. 4.7. In this study, the parameter h_m is set as a constant because a large change in h_m can destabilize the robot's posture, whereas w_m is varied. . For example, in case of the loss of two legs, we can adjust the parameter w_m of two legs to increase the size of the trajectory loop along the horizontal direction. If the parameters w_m are set diversely, the robot can be maneuvered to turn left or right. However, it is difficult to tune the parameters. In the next section, we integrate the mudskipper-inspired movement with reinforcement learning to improve the robot with the flexibility to perform in various scenarios.

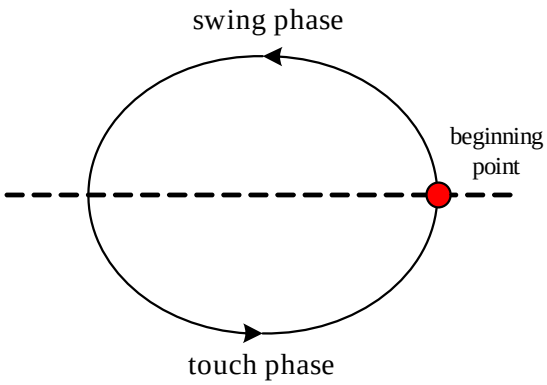


Figure 4.6: Moving trajectory of robot's leg.

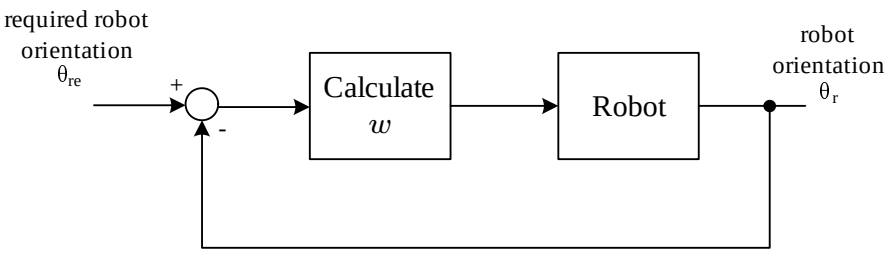


Figure 4.7: Feedback control diagram for mudskipper-inspired movement.

4.3.2 Q-Learning Algorithm

Reinforcement learning can be simply explained as the process of training pets, such as dogs and cats, to perform certain actions. However, this process can be described as a trial-and-error problem as well. In the beginning of a dog's training process, it seems impossible for a dog to perform a requested task without error. The dog will try to perform some action randomly, such as walking, sitting, or jumping, to get a snack. After it gets the snack, the dog will remember the action that can possibly get it a snack again. By contrast, the dog will not do any action that would lead to an adverse outcome such as a scolding or a beating. Accordingly, the process of reinforcement learning can be thought to be based on reward and punishment. For example, Q-learning with fuzzy reward was developed to control the hexapod robot [72]. It was also employed to control the humanoid robot as well for enhancing adaptation [88, 98]. In this study, the quadruped robot is controlled using Q-learning and mudskipper-inspired behavior. The control policy will be adapted with the current state, provided by the surrounding environment, following Q-learning approach. For example, if the robot is in a state in which it is heading to the north direction, but we command it to go to the to the east direction, the control policy will be changed to make the robot go to the east direction. The three main parameters of reinforcement learning are as follows:

- State (S): It is defined as the current scenario of a system, for instance, the position of the robot.
- Action (A): In one system, several actions would be required to be conducted in each state. In wheel-based robots, the actions can be moving forward, turning left, and turning right.
- Reward (R): It depends on the current state and action. It can be positive, negative, or zero for the win, lose, and draw scenarios, respectively. For example, when a robot encounters an obstacle, it needs to avoid the obstacle. If the robot decides to move forward and hit an obstacle, it will get a negative outcome in the form of a punishment. On the contrary, if the robot avoids an obstacle properly, it will receive a positive reward.

Finally, the objective of learning, which is known as policy (π), will be achieved. The best policy is the selection of actions that provide the highest reward, the so-called "Maximum Sum of Expected Rewards".

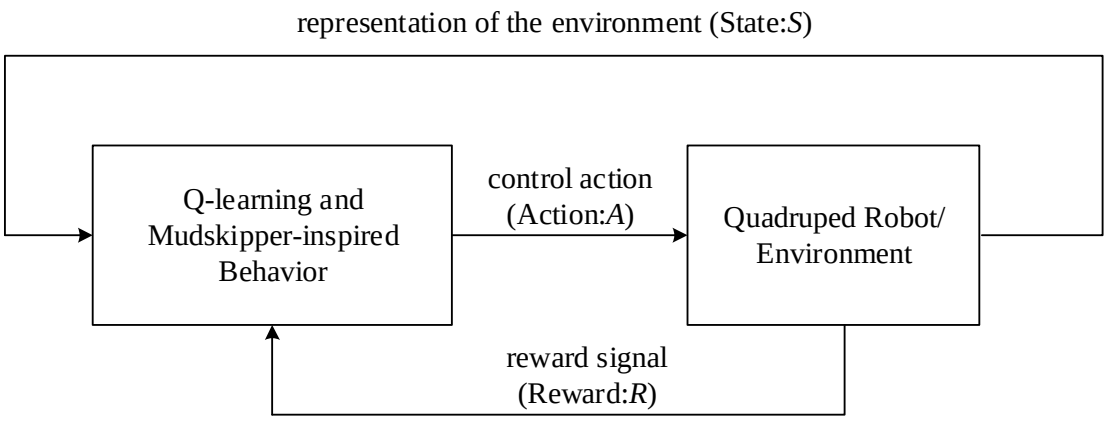


Figure 4.8: Controller structure for proposed method (SLMIC).

Q-learning is a reinforcement learning approach with non-model requirement. It employs the concept of Markov decision process (MDP) with finite state arrays to arrive at the optimal policy [8]. The expected reward will be stored in a d-dimensional state-action array. The number of d can be set manually by the user. Q-learning is one algorithm among the various algorithms associated with the temporal-difference method. The Q-values are acquired as follows:

$$V^*(s) = \max_a Q(s, a) \tag{4.7}$$

Because the Q-function does not need a model for learning and selecting actions, no model is required for state transitions. Q is updated by using the new information obtained after performing an action to correct the old policy. The Q-values are updated numerically based on the temporal-difference concept using Eq.(4.8) [67].

$$Q(s, a) = Q(s, a) + \alpha(R(s) + \gamma \max_{a'} Q(s', a') - Q(s, a)) \tag{4.8}$$

where α and γ are the learning rate and the discount factor, respectively. The procedure of the Q-algorithm is summarized in Table 4.1. The proposed method employs both Q-learning and mudskipper-inspired behavior to control the damaged robot. The controller structure is illustrated in Fig. 4.8.

Table 4.1: Q-learning procedure for n^{th} episode.

Algorithm: Q-learning Algorithm	
1:	observes its current state s_n.
2:	selects and perform an action a_n.
3:	observes the subsequent state s'_n.
4:	receives an immediate reward r_n.
5:	adjust it Q_{n-1} value using Eq. (4.8)

4.3.3 Learning Mudskipper Moving Algorithm

To increase the performance of Mudskipper-inspired movement, the learning approach is integrated to let the robot move in the straight direction. It has been decades that learning algorithm was combined with robots to improve the flexibility and efficiency. The reinforcement learning was utilized in many application in fields of robotics. The advantage of it is the self-learning scenario. In [96], the wheel-based robot was programmed to be able to learn online. As well as high DOFs robots, the humanoid robot can learn how to walk and crawling successfully in [52][97]. Moreover, the speed of walking can be improved with the self-learning quadruped robot reported in [48]. Although these researches were conducted successfully with reinforcement learning, a key problem with much of the learning approach is the size of state and action space. Generally, for multi-joint robot, the state is set as the current position of joint angles so that the state will become 12 dimension for this study – due to the number of DOFs of quadruped robot. Thus, the new concept of state and action design is proposed in this study.

State

In spite of using joint angles, we employ robot orientation as the state in the Q-learning. The yaw angle of the robot is divided into 7 regions, as shown in Fig. 4.9. The regions that separate the state are listed in Table 4.2. We select the robot orientations as the state to be able to control robot direction properly, and another advantage of doing so is that a robot can learn other actions simultaneously if provided with another Q-value table.

Table 4.2: The design space region and state rewards.

State	Region	Reward R_s
S0	$\theta_r < -25$	-10
S1	$-25 \leq \theta_r < -15$	-5
S2	$-15 \leq \theta_r < -5$	-1
S3	$-5 \leq \theta_r < 5$	0
S4	$5 \leq \theta_r < 15$	-1
S5	$15 \leq \theta_r < 25$	-5
S6	$25 \leq \theta_r$	-10

Action

Because the mudskipper-inspired movement requires only two legs to realize crawling, the number of robot actions is set to 7 for both legs by tuning the parameter w_m in Eq. (4.5). The parameters w_{m1} and w_{m2} are used to control the legs L_1 and L_2 as shown in Fig. 4.9, and w_{m1} and w_{m2} are calculated using the following equations.

$$w_{m1} = 5 + a_1 \tag{4.9}$$

$$w_{m2} = 5 + a_2 \tag{4.10}$$

where a_1 and a_2 are the adjusting parameters, and their values are set as given in Table 4.3. The numbers used for each action are set unequally to ensure the robot perform different actions, such as turn left and turn right. Therefore, the total number of Q-value is $7 \times 7 = 49$, which is smaller than that in the conventional setting.

Reward

Because the aim of the present study is to design a recovery method to ensure that a robot can move after sustaining damage, we set the reward as the summation of two rewards, namely, state reward (R_s) and action reward (R_a).

$$R = R_a + R_s \tag{4.11}$$

As a result, we can control not only the robot orientation but also the movability, that is, distance traveled. To facilitate robot motion in straight line, the

Table 4.3: Design action set.

Action	a_1 (Leg L_1)	a_2 (Leg L_2)
A0	4	1
A1	3	1
A2	2	1
A3	1	1
A4	1	2
A5	1	3
A6	1	4

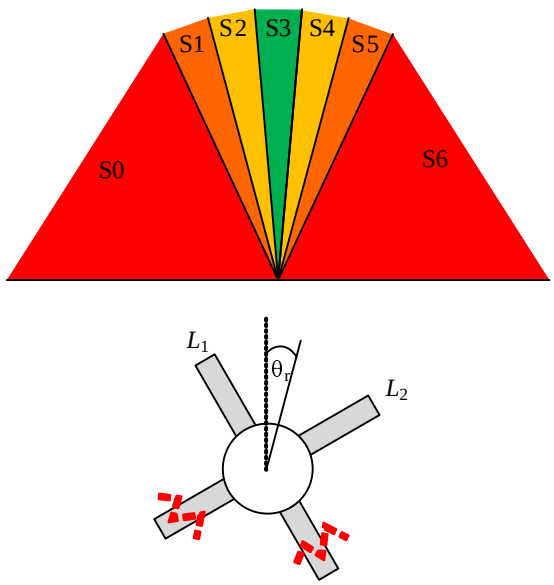


Figure 4.9: State space for Q-algorithm of the robot orientation θ_r.

state reward was designed to be the values listed in Table 4.2. The robot gets the negative reward (punishment) if it turns left or right. This ensures that after leaning, the robot moves in the desired direction. Furthermore, the action reward is used to propel the robot to move forward faster, as expressed by Eq. (4.12). If the robot moves backward, it will be punished with -10 reward. By contrast, if the robot can move the longest distance, it will receive +10 reward.

$$R_a = \begin{cases} 10 & \text{if max distance} \\ d & \\ -10 & \text{if moving backward} \end{cases} \tag{4.12}$$

where d is the distance traveled by the robot.

4.4 Result and Discussion

To test the performance of the proposed method, the experiment was conducted in both simulation and actual robot. The broken robot used in this experiment consist of two healthy legs and two legs ripped out. The robot use the IMU sensor to measure the orientation. In the simulation, the traveled distance can be calculated by the absolute position of the robot provided by the simulation software. However, it seems impossible to get the correct position of real robot in the practical experiment so that the short-time distance provided by IMU is employed instead.

4.4.1 Evaluation of Self-learning Mudskipper-inspired Crawling Method (SLMIC)

In the benchmark, the proposed recovery algorithm was tested in four different scenarios and compared to the conventional evolutionary method without direction (EA), evolutionary method with direction (EAD) and mudskipper-inspired movement (MUD), in the simulation. The test scenarios were as follows:

- case A: one leg lost,
- case B: two adjacent legs lost,
- case C: two diagonal legs lost,
- case D: two adjacent legs and one limb lost.

All the simulation scenarios were run for 20 s with the four methods. After the simulations were executed, only the proposed method, that is, SLMIC, was employed to operate the actual robot because the evolutionary processes were time consuming and SLMIC provided the best performance in terms of recovery distance. The performance of SLMIC was compared with the control method used to control the robot before it was damaged. To achieve the robot position, Inertial Measurement Unit (IMU) was incorporated into the robot and was used to measure the short distance that was required in the recovery process. Image processing was additionally used to obtain the distance traveled by the robot for reporting purposes.

4.4.2 Simulation Results of SLMIC vis-a-vis Other Methods

The numerical simulation was conducted separately for each case in the experiments. As shown in Fig. 4.10, the results of four different damage scenarios are as follows:

1. case A: The results show that all methods used in the simulation allowed the robot to be able to move again. With one leg lost, it was easy for the robot to travel with three functional legs. However, not all method provided the acceptable results. EA help the robot move the shortest distance compared with other methods, as shown in Fig. 4.10(b). EAD help the robot move longer than EA but it lost out to the MUD and SLMIC methods. By employing the specific actions of mudskipper, both MUD and SLMIC help the robot travel longer distances. However, SLMIC provided the best result in this test because it helped the robot learn to move forward faster.

2. case B: The robot programmed using EA traveled faster than the robot programmed using EAD, as shown in Fig. 4.10(c). However, EAD provided the better result in terms of direction. It seemed that, with EAD, the robot optimized multiple objectives, namely, distance traveled and direction of travel. As a result, the robot assigned more importance to direction in optimization, which reduced the distance traveled. MUD and SLMIC provided decent results in terms of distance traveled and direction of travel. Once again, SLMIC provided the best performance.

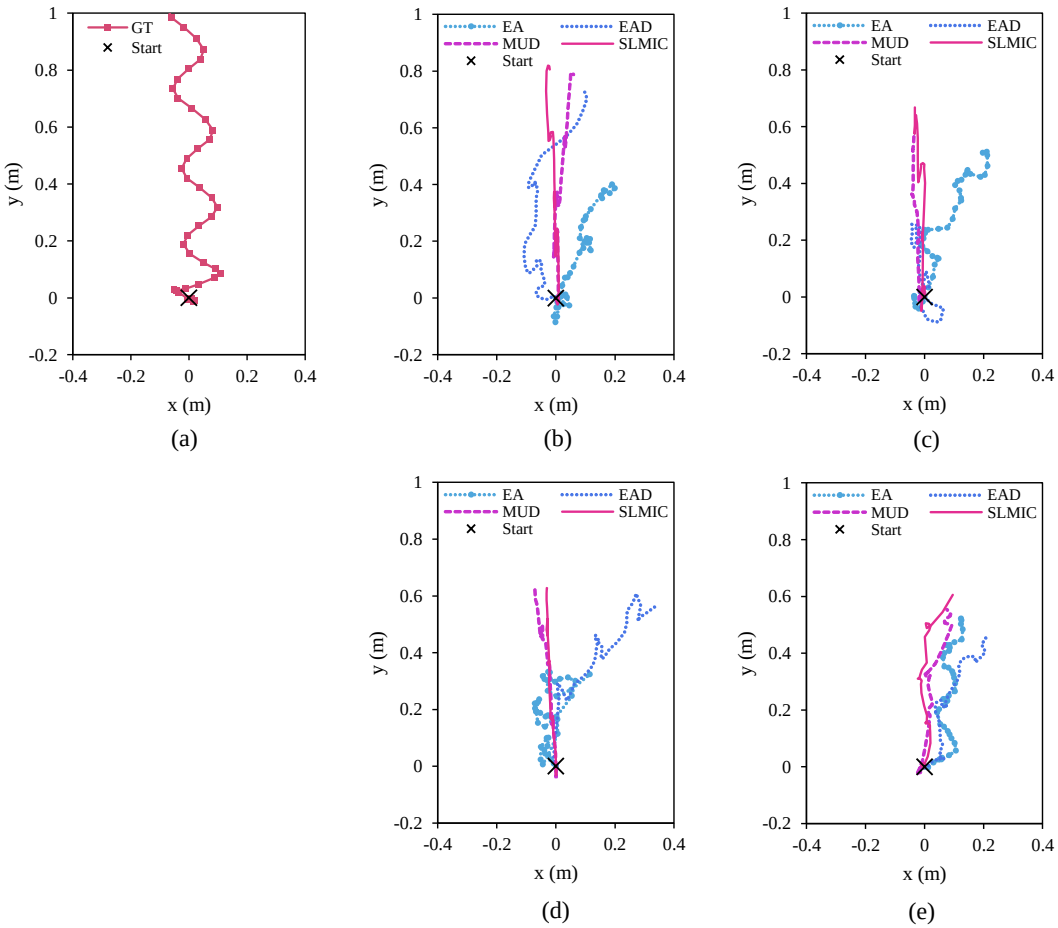


Figure 4.10: Comparison of simulation results obtained using EA, EAD, MUD and SLMIC: (a) ground truth of healthy robot walking with trotting gait; (b) one leg lost (case A); (c) two adjacent legs lost (case B); (d) two diagonal legs lost (case C); (e) two adjacent legs and one limb lost (case D).

3. case C: Similar to the two cases in the experiments, with MUD and SLMIC the robot covered longer distances. However, SLMIC performed better in terms of direction of travel. Opposite to case B, EAD could deal with only the distance traveled. At this time, EAD attempted to optimize the distance traveled by the robot, but it failed to optimize the direction of travel, and thus, the robot failed to move straight ahead. With EA, the robot could not perform well because the two diagonal legs affected its balance. The robot flipped over during the recovery process which limited its ability to move. As a result, the robot programmed using EA could travel properly, as shown in Fig. 4.10(d).

4. case D: This experiment was the most challenging because of the limited number of functional legs and actuators, as shown in the results in Fig. 4.10(e). Given the extremities, the robot programmed using SLMIC could learn to recovery itself with SLMIC and provided the best results in terms of direction and distance. MUD with its specific control method was second best performer in this experiment. EAD exhibited the worst performance owning to the same reason as in case B, and EA achieved a fair level of performance.

From the results, it can be concluded that the proposed method (SLMIC) can provide the best results compared to the other methods in terms of direction of travel and the distance traveled. However, a certain learning time is required to achieve the goal. The performance of MUD was inferior to that of SLMIC, but, in most cases, it performed satisfactorily. However, the crucial aspect of MUD is that it employs specific mudskipper-inspired actions with no time required for evolutionary process or learning. The results indicate that it would be difficult to guarantee satisfactory performance in terms of direction and distance traveled with both EA and EAD, because differences in robot model can cause task failures, and the methods can get stuck in local minima when performing multi-objectives optimization, which can lead to failure.

4.4.3 Comparison of Experimental Results Obtained using Previous Control Method and SLMIC

Becuase SLMIC exhibited the best performance in the simulation, we decided to conduct the experiment involving the actual robot using only with SLMIC. We

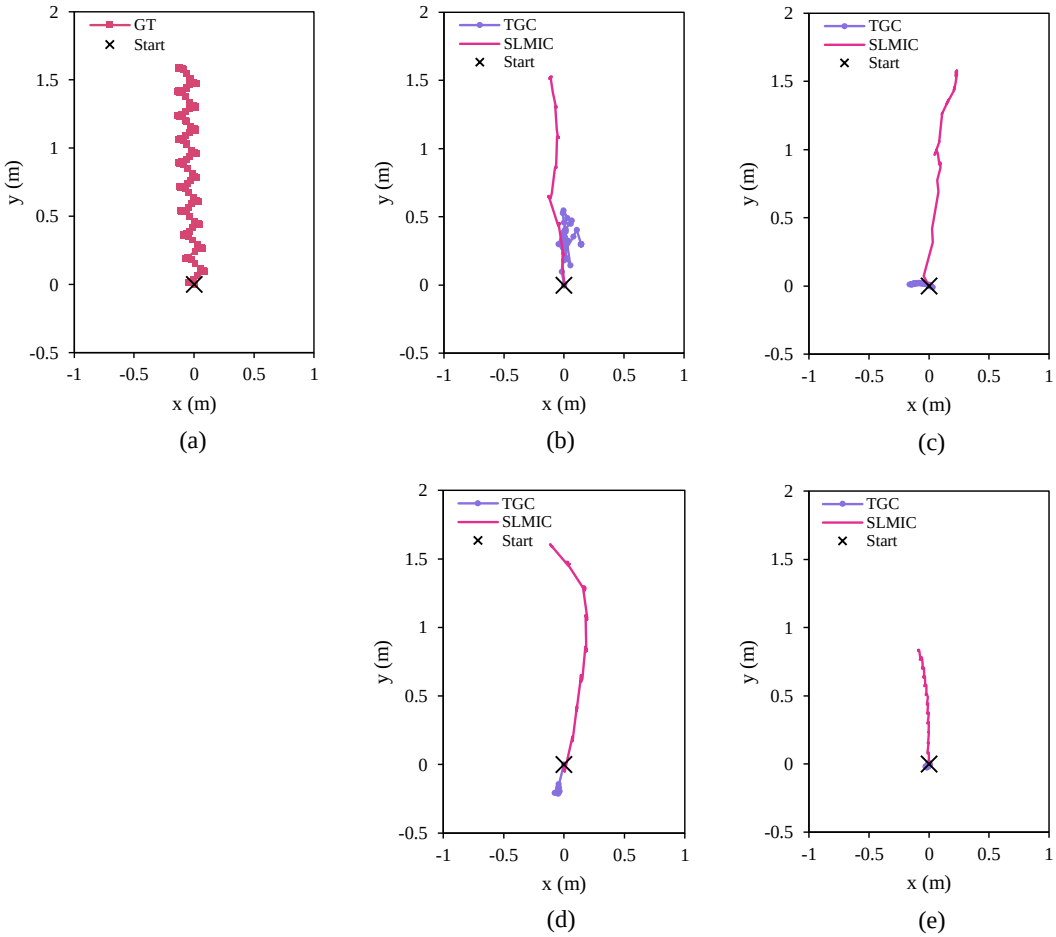


Figure 4.11: Experimental results of SLMIC compared to previous controller (trotting gait): (a) ground truth of healthy robot walking with trotting gait; (b) one leg lost (case A); (c) two adjacent legs lost (case B); (d) two diagonal legs lost (case C); (e) two adjacent legs and one limb lost.

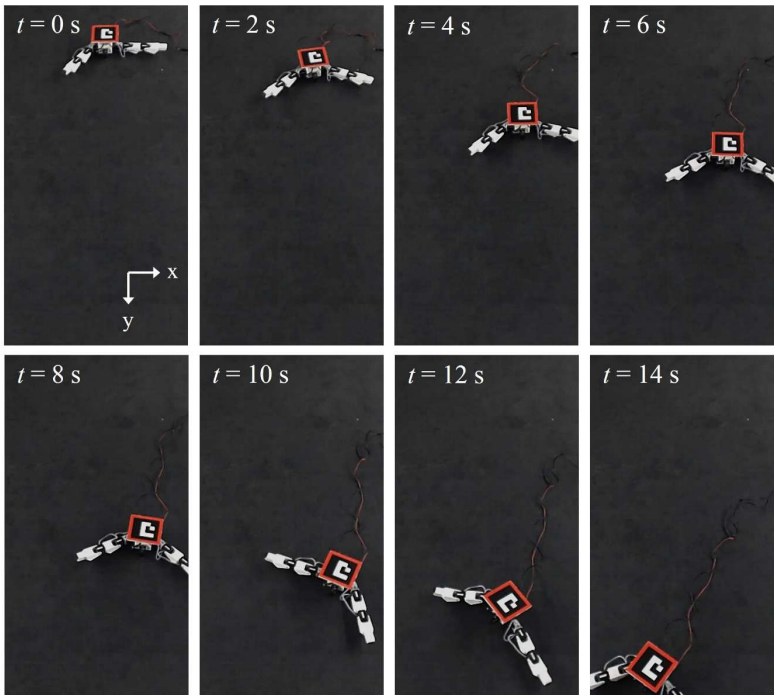


Figure 4.12: CIQR with two diagonal legs performing recovery action after learning with SLMIC method.

compared the performance of SLMIC with the previous controller, that is, trotting gait (TG). The test cases were same as those in the experiments conducted in section 4.2.1.

1. case A: The results of this test case clearly show that with SLMIC the robot could recover itself to reach the goal, as shown in Fig. 4.11(b). Compared to the ground truth (in Fig. 4.11(a)), the robot programmed with SLMIC almost traveled the same distance as the healthy robot. By contrast, the robot could not move well when the previous control method (trotting gait controller) was employed. It caused the robot to move back and fourth around a single point in a certain working area.

2. case B: In this case, SLMIC with the damaged robot achieved the same result as the healthy robot. However, the robot moved slightly towards the right part of the working space. By contrast, the robot with two adjacent legs-lost and programmed by the previous method could not perform well, traveling only around the starting point, as shown in Fig. 4.11(c).

3. case C: As shown in Fig. 4.11(d), with the trotting gait, the broken robot could not function properly and moved backwards during the experiment. This can be one of the reason why the recovery method is significant for multiple-legged robots. By contrast, the proposed method provided good performance with the learning process. According to the trajectory traveled by the robot programmed with the proposed method, it moved towards the right at the beginning, but it returned to the predetermined direction with the passage of time.

4. case D: Similar to results of the simulation in the previous section, the robot with two adjacent legs and one limb lost found it difficult to achieve the same performance as the healthy robot. However, SLMIC made a big difference compared to the previous controller. Even so, it could not help the robot recover fully, but it did help the damaged robot cover more than half the distance covered by the healthy robot.

After the experiments with actual robot were conducted properly, it was found that the damaged robot could practically not move when previous controller (trotting gait in this case) was used. As a result, the recovery method that can provide an alternative solution for damaged robots is needed. With SLMIC, the proposed

recovery method based on specific actions that guarantee adaptable movement owning to learning could solve most of the problems successfully, and in one special difficult case, it recovered by approximately 50 percent. The example of robot actions with SLMIC in Fig. 4.12 (case C) shows that the robot turned slightly to the wrong direction at the beginning but it recovered and moved along the correct direction at the end. Finally, the experiments confirmed that the proposed method (SLMIC) is suitable for quadruped robot with a limited number of functional legs.

4.5 Summary

In this chapter, several techniques of a self-recovery method for legged robot have been discussed in the beginning. The evolutionary-based recovery method with PSO algorithm has been explained in detail how the robot discovers the new, alternative solution to get recovery from damage. Further, the mudskipper-inspired method is detailed. The extraordinary locomotion of mudskipper is defined as the 2-DOFs system and used to control the robot with two legs. The modified version of the mudskipper-inspired method with Q-learning algorithm have been proposed, namely, self-learning mudskipper-inspired crawling method (SLMIC). The novel setting of reinforcement learning for the multi-DOFs system is investigated. Finally, the simulation and experimental results show that the proposed method provides the best result in terms of direct control. The robot can move faster after learning with the mudskipper movement. Our work has led us to the conclusion that the proposed method is feasible to operate with a damaged robot that has two legs.

Chapter 5
Conclusion and Future Work

5.1 Conclusion

In conclusion, this study mainly focuses on the development of the self-recovery strategy for the legged robot. It is noted that only the servo motor-based legged robots are considered in this research. Leg loss and joint lock problems are set as the breakdown of robots, which are intensively inspected to solve. We have found novel solutions to compensate for the faults of legged robots based on the bio-inspired aspects. A new quadruped robots' structure has been proposed. Moreover, the development of self-fault-detection algorithms is included in this thesis. Last, by not least, the reinforcement learning-based recovery method is proposed as well.

The contributions of this study are listed as follows:

- Caterpillar-inspired Quadruped Robot (CIQR): This algorithm is developed in this work to improve the ability to move when the robot has malfunction parts. The caterpillar's proleg is added on the robot limb as a triangular shape. The optimization process is therefore conducted to find the optimal solution for the height of a proleg.

- PSO-based Leg-loss Identification method (PLI): To get the new model of the damaged robot, PSO algorithm and robot's orientation are acted as the main operator. The new, updated model can be achieved by comparing the feedback signals between the candidate models in the simulation and the actually damaged robot in a real environment.

- Acoustic-based fault diagnosis for legged robots (AFL): AFL is proposed to decrease the number of sensors utilized on the robot. With the sound of motors, the robot can diagnosis the errors of joint faults. The FFT and fuzzy classification are used to process the acoustic signals.

- Self-learning mudskipper-inspired crawling method (SLMIC): The simple concept of mudskipper locomotion is applied to the damaged multi-legged robot in order to control it to the target direction. In addition, Q-learning, a reinforcement learning method, is combined with mudskipper's behavior to let the robot move faster in a desired direction.

5.2 Future Work

According to section 1.3, this field of research quite challenges. There are some gap and limitation of this study to be improved and continued. The research problems that can be solved further are detailed as below:

- Different design of robot structure: In this study, we focused only on a triangular shaped of proleg with a limited number of parameters, and the optimization process is set as a one-objective problem. For future work, we suggest designing a new shape of the limb to improve the performance and apply a multi-objective optimization method to get better results. It is recommended to consider the force between the robot and surface to get the practical solution in the simulation.

- Transferability between simulation and actual robots: Since in this field, simulation is the main tool to minimize the processing time, the lask between real world and simulation will cause a pitfall. To make the simulation unite with the real world successfully, the transferring module is needed.

- Fault detection technique: Due to the fact that only two faults are studied in this work, on a wider level, research is also needed to determine future solutions to cope other failures of the legged robot system.

- Improvement of learning method: As this study focuses on a legged robot with a small number of active legs, testing with several kinds of a legged robot is a fundamental issue for future research.

Bibliography

[1] S. Abe and M.-S. Lan. A method for fuzzy rules extraction directly from numerical data and its application to pattern classification. *Fuzzy Systems, IEEE Transactions on*, 3(1):18–28, Feb 1995.

[2] A. Abid and M. T. Khan. Multi-sensor, multi-level data fusion and behavioral analysis based fault detection and isolation in mobile robots. In *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 40–45, Oct 2017.

[3] M. Ahmadi and M. Buehler. The arl monopod ii running robot: control and energetics. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 3, pages 1689–1694 vol.3, May 1999.

[4] D. Ahn and B. Cho. Optimal periodic hopping trajectory generation for legged robots. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1263–1268, Jul 2018.

[5] M. M. U. Atique and M. A. R. Ahad. Inverse kinematics solution for a 3dof robotic structure using denavit-hartenberg convention. In *2014 International Conference on Informatics, Electronics Vision (ICIEV)*, pages 1–5, May 2014.

[6] H. Azgomi and J. Poshtan. Induction motor stator fault detection via fuzzy logic. In *the 21st Iranian Conference on Electrical Engineering (ICEE)*, pages 1–5, May 2013.

[7] T. Barfoot, E. Earon, and G. D'Eleuterio. Experiments in learning distributed control for a hexapod robot. *Robotics and Autonomous Systems*, 54(10):864 – 872, 2006.

[8] R. Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[9] S. Chattunyakit, Y. Kobayashi, and T. Emaru. Joint fault diagnosis of legged robot based on acoustic processing. In *2015 IEEE/SICE International Symposium on System Integration (SII)*, pages 169–174, Dec 2015.

[10] S. Chattunyakit, Y. Kobayashi, and T. Emaru. Pso-based leg-loss identification method for legged robots. *Journal of Research and Applications in Mechanical Engineering*, 4(1):46–57, 2016.

[11] S. Chattunyakit, Y. Kobayashi, T. Emaru, and A. A. Ravankar. Bio-inspired structure and behavior of self-recovery quadruped robot with a limited number of functional legs. *Applied Sciences*, 9(4), 2019.

[12] L. Chen, G. Zhong, Z. Liu, and H. Deng. A novel hexapod robot: Design and mobility analysis. In *2016 IEEE International Conference on Mechatronics and Automation*, pages 2094–2098, Aug 2016.

[13] X. Chen, C. Qi, F. Gao, X. Tian, X. Zhao, and H. Yu. Fault-tolerant gait a quadruped robot with partially fault legs. In *2014 UKACC International Conference on Control (CONTROL)*, pages 509–514, Jul 2014.

[14] S. K. . Chu and G. K. . Pang. Comparison between different model of hexapod robot in fault-tolerant gait. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 32(6):752–756, Nov 2002.

[15] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 512:503–507, 2015.

[16] O. Darici, M. K. Yalcin, and H. Temeltas. Comparison of gait generation methods in quadruped walking. In *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1–6, Jul 2008.

[17] B. R. de Oliveira Floriano, C. de Freitas Porphirio, P. H. Moreira Santana, G. Araújo Borges, and A. R. Soares Romariz. Walking pattern design and balance control of a quadruped platform. In *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, pages 242–247, Nov 2018.

[18] K. Deb and D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[19] H. Du and F. Gao. Fault tolerance properties and motion planning of a six-legged robot with multiple faults. *Robotica*, 35(6):1397–1414, 2017.

[20] Z. Duan, Z. Cai, and J. Yu. Unknown fault detection for mobile robots based on particle filters. In *2006 6th World Congress on Intelligent Control and Automation*, volume 2, pages 5452–5456, Jun 2006.

[21] Z. Duan, H. Ma, and L. Yang. Fault detection for internal sensors of mobile robots based on support vector data description. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pages 2702–2706, May 2015.

[22] E. Dubrova. *Fault-Tolerant Design*. Springer Publishing Company, Incorporated, 2013.

[23] S. Faraji, S. Pouya, R. Moeckel, and A. J. Ijspeert. Compliant and adaptive control of a planar monopod hopper in rough terrain. In *2013 IEEE International Conference on Robotics and Automation*, pages 4818–4825, May 2013.

[24] S. Floyd, T. Keegan, J. Palmisano, and M. Sitti. A novel water running robot inspired by basilisk lizards. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5430–5436, Oct 2006.

[25] M. Fujita. Aibo: Toward the era of digital creatures. *The International Journal of Robotics Research*, 20(10):781–794, 2001.

[26] H. Gao, T. Wang, J. Liang, and Y. Zhou. Model adaptive gait scheme based on evolutionary algorithm. In *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, pages 316–321, Jun 2013.

[27] Z. Gao, L. Ma, and J. Wang. Fault tolerant control method for displacement sensor fault of wheel-legged robot based on deep learning. In *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, pages 147–152, Aug 2018.

[28] Y. Geva and A. Shapiro. A novel design of a quadruped robot for research purposes. *International Journal of Advanced Robotic Systems*, 11(7):95, 2014.

[29] H. Ghaderi and P. Kabiri. Fourier transform and correlation-based feature selection for fault detection of automobile engines. In *the 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*, pages 514–519, May 2012.

[30] P. González-de Santos, E. Garcia, and J. Estremera. *Quadrupedal Locomotion: An Introduction to the Control of Four-legged Robots*. Springer-Verlag, Berlin, Heidelberg, 2006.

[31] Y. Guangyou. A modified particle swarm optimizer algorithm. In *2007 8th International Conference on Electronic Measurement and Instruments*, pages 2–675–2–679, Aug 2007.

[32] F. Guenther and F. Iida. Energy-efficient monopod running with a large payload based on open-loop parallel elastic actuation. *IEEE Transactions on Robotics*, 33(1):102–113, Feb 2017.

[33] K. Harbick and G. S. Sukhatme. Controlling hopping height of a pneumatic monopod. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 4, pages 3998–4003 vol.4, May 2002.

[34] I. Hashlamon and K. Erbatur. Joint sensor fault detection and recovery based on virtual sensor for walking legged robots. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 1210–1214, Jun 2014.

[35] J. Heaston, D. Hong, I. Morazzani, P. Ren, and G. Goldman. Strider: Self-excited tripedal dynamic experimental robot. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2776–2777, Apr 2007.

[36] R. Hirose and T. Takenaka. Development of the humanoid robot asimo. *Honda R&D Technical Review*, 13:1–6, Jan 2001.

[37] K. Hosoda, H. Takayama, and T. Takuma. Bouncing monopod with biomimetic muscular-skeleton system. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3083–3088, Sep. 2008.

[38] A. Jamal and N. Verma. Automatic fault diagnosis system using acoustic data. In *the 8th IEEE International Conference on Industrial and Information Systems (ICIIS)*, pages 421–426, Dec 2013.

[39] J. Jara and R. Acevedo-Crespo. Crisp classifiers vs. fuzzy classifiers: A statistical study. In M. Kolehmainen, P. Toivanen, and B. Beliczynski, editors, *Adaptive and Natural Computing Algorithms*, volume 5495 of *Lecture Notes in Computer Science*, pages 440–447. Springer Berlin Heidelberg, 2009.

[40] D. Jaramillo and C. Narvaez. Vehicle online monitoring system based on fuzzy classifier. In *the 3rd International Conference on Advances in Vehicular Systems, Technologies and Applications*, pages 33–38, Jun 2014.

[41] A. M. Johnson, G. C. Haynes, and D. E. Koditschek. Disturbance detection, identification, and recovery by gait transition in legged robots. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5347–5353, Oct 2010.

[42] W. Jutharee and T. Maneewarn. Gesture reconfiguration from joint failure using genetic algorithm. In *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, pages 1137–1142, Oct 2016.

[43] K. Kaneko, M. Morisawa, S. Kajita, S. Nakaoka, T. Sakaguchi, R. Cisneros, and F. Kanehiro. Humanoid robot hrp-2kai — improvement of hrp-2 towards disaster response tasks. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 132–139, Nov 2015.

[44] D. Kim, J. I. Kim, and Y. Park. A simple tripod mobile robot using soft membrane vibration actuators. *IEEE Robotics and Automation Letters*, 4(3):2289–2295, Jul 2019.

[45] J. Kim, H. Kim, Y. Kim, H. S. Kim, and J. Kim. Design of lizard-inspired robot with lateral body motion. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, Oct 2018.

[46] J. T. Kim, J. S. Cho, B. Y. Park, and S. Park. Inverse kinematics for control of jinpoong. In *2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 470–472, Oct 2013.

[47] H. Kimura, T. Yamashita, and S. Kobayashi. Reinforcement learning of walking behavior for a four-legged robot. 122:330–337, Mar 2002.

[48] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 3, pages 2619–2624 Vol.3, Apr 2004.

[49] S. Koos, A. Cully, and J.-B. Mouret. Fast damage recovery in robotics with the t-resilience algorithm. *The International Journal of Robotics Research*, 32(14):1700–1723, 2013.

[50] Y. Li, B. Li, J. Ruan, and X. Rong. Research of mammal bionic quadruped robots: A review. In *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*, pages 166–171, Sep. 2011.

[51] J. Liang and C. Xue. Self identification and control of four-leg robot based on biological evolutionary mechanisms. In *2010 5th IEEE Conference on Industrial Electronics and Applications*, pages 958–961, Jun 2010.

[52] J. L. Lin, K. S. Hwang, W. C. Jiang, and Y. J. Chen. Gait balance and acceleration of a biped robot based on q-learning. *IEEE Access*, 4:2439–2449, 2016.

[53] M. Liu, M. Li, and J. Pang. Fault-tolerant gait implementation of hexapod robot based on finite state automata. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 6800–6805, May 2017.

[54] H. Matsuoka Medeiros, W. Tanaka Botelho, F. Marin Biihrer, D. de Sousa Rodrigues, and M. d. G. Bruno Marietto. Transition to paw/hoof in bio-inspired quadruped robot tb-horse ii and its demonstrative simulation. In *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, pages 236–241, Nov 2018.

[55] B. McInroe, H. C. Astley, C. Gong, S. M. Kawano, P. E. Schiebel, J. M. Rieser, H. Choset, R. W. Blob, and D. I. Goldman. Tail use improves performance on soft substrates in models of early vertebrate land locomotors. *Science*, 353(6295):154–158, 2016.

[56] M. Mladenov, M. Mock, and K.-E. Grosspietsch. Fault monitoring and correction in a walking robot using lms filters. In *the International Workshop on Intelligent Solutions in Embedded Systems*, pages 1–10, Jul 2008.

[57] M. Murakami. Task-based dynamic fault tolerance for humanoid robots. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2197–2202, Oct 2006.

[58] M. Nandhini, V. Krithika, and K. Chittal. Design of four pedal quadruped robot. In *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, pages 2548–2552, Sep 2017.

[59] K. O. Omali, M. Benbrahim, and M. N. Kabbaj. Fault detection and isolation for manipulator robot. In *2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, pages 1–5, Nov 2018.

[60] S. Parasuraman and V. L. Jauw. Quad crawler robot navigation using animal walking gait. In *2010 2nd International Conference on Electronic Computer Technology*, pages 232–237, May 2010.

[61] I. Park, J. Kim, and J. Oh. Online biped walking pattern generation for humanoid robot khr-3(kaist humanoid robot - 3: Hubo). In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 398–403, Dec 2006.

[62] S. Peng, C. P. Lam, and G. R. Cole. A biologically inspired four legged walking robot. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, pages 2024–2030 vol.2, Sep. 2003.

[63] G. Y. Qiu and S. H. Wu. The evolutionary locomotion of tripedal and quadrupedal biomorphic robots. In *2011 International Conference on Technologies and Applications of Artificial Intelligence*, pages 45–50, Nov 2011.

[64] K. R. Rao and P. Yip, editors. *The Transform and Data Compression Handbook*. CRC Press, Inc., Boca Raton, FL, USA, 2000.

[65] G. Ren, W. Chen, S. Dasgupta, C. Kolodziejski, F. Wörgötter, and P. Manoonpong. Multiple chaotic central pattern generators with learning

for legged locomotion and malfunction compensation. *Information Sciences*, 294:666 – 682, 2015. Innovative Applications of Artificial Neural Networks in Engineering.

[66] A. Roennau, G. Heppner, T. Kerscher, and R. Dillmann. Fault diagnosis and system status monitoring for a six-legged walking robot. In *the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 874–879, Jul 2011.

[67] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artifi. Prentice Hall, 2010.

[68] F. Salazar-Villanueva and O. Ibarra-Manzano. Spectral analysis for identifying faults in induction motors by means of sound. In *the International Conference on Electronics, Communications and Computing (CONIELECOMP)*, pages 149–153, Mar 2013.

[69] D. Sarkar, S. K. Dubey, A. Mahapatra, and S. S. Roy. Modeling and analysis of fault tolerant gait of a multi-legged robot moving on an inclined plane. *Procedia Technology*, 14:93 – 99, 2014. 2nd International Conference on Innovations in Automation and Mechatronics Engineering, ICIAME 2014.

[70] F. Seljanko. Fault detection algorithm for legged walking robot. In *the IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1211–1216, Aug 2013.

[71] J. Seo, J. Cho, B. Park, J. Kim, and S. Park. Leg mechanism design for slip model of hydraulic quadruped robot. In *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 461–466, Nov 2014.

[72] M. Shahriari and A. A. Khayyat. Gait analysis of a six-legged walking robot using fuzzy reward reinforcement learning. In *2013 13th Iranian Conference on Fuzzy Systems (IFSC)*, pages 1–4, Aug 2013.

[73] Y. Shatnawi and M. Al-khassaweneh. Fault diagnosis in internal combustion engines using extension neural network. *Industrial Electronics, IEEE Transactions on*, 61(3):1434–1443, Mar 2014.

[74] Z. Shuai, Q. Yanhuai, and H. Qingkai. Fault characteristics analysis for rotor systems with misalignment based on wavelet packet decomposition and frequency-band energy ratio analysis. In *the 30th Chinese Control Conference (CCC)*, pages 4331–4335, Jul 2011.

[75] M. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2005.

[76] Y. Stein. *Digital Signal Processing: A Computer Science Perspective*. Oct 2000.

[77] F. Sun, H. H. Ju, and P. Y. Cui. A new 12 dof biped robot's mechanical design and kinematic analysis. In *Proceedings of 2011 International Conference on Electronic Mechanical Engineering and Information Technology*, volume 5, pages 2396–2400, Aug 2011.

[78] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[79] H. Suzuki, J. H. Lee, and S. Okamoto. Development of semi-passive biped walking robot embedded with cpg-based locomotion control. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 75–78, Jun 2017.

[80] L. Swathy and L. Abraham. Analysis of vibration and acoustic sensors for machine health monitoring and its wireless implementation for low cost space applications. In *the First International Conference on Computational Systems and Communications (ICCSC)*, pages 80–85, Dec 2014.

[81] F. Tedeschi and G. Carbone. Design issues for hexapod walking robots. *Robotics*, 3:181–206, Jun 2014.

[82] R. Tedrake, T. Weirui Zhang, and H. Sebastian Seung. Learning to walk in 20 minutes. In *Proceedings of Yale workshop on Adaptive and Learning Systems*, Sep 2005.

[83] J. Tenreiro Machado and M. Silva. An overview of legged robots. Apr 2006.

[84] B. A. Trimmer, A. E. Takesian, B. M. Sweet, C. B. Rogers, D. C. Hake, and D. J. Rogers. Caterpillar locomotion: A new model for soft- bodied climbing and burrowing robots. 2007.

[85] C. T. Trung, H. M. Son, D. P. Nam, T. N. Long, D. T. Toi, and P. A. Viet. Fault detection and isolation for robot manipulator using statistics. In *2017 International Conference on System Science and Engineering (ICSSE)*, pages 340–343, Jul 2017.

[86] I. Tsitsimpelis, C. J. Taylor, B. Lennox, and M. J. Joyce. A review of ground-based robotic systems for the characterization of nuclear environments. *Progress in Nuclear Energy*, 111:109 – 124, 2019.

[87] K. S. Tso and P. G. Backes. A fail-safe tele-autonomous robotic system for nuclear facilities. *Robotics and Computer-Integrated Manufacturing*, 10(6):423 – 428, 1993.

[88] C. M. C. O. Valle, R. Tanscheit, and L. A. F. Mendoza. Computed-torque control of a simulated bipedal robot with locomotion by reinforcement learning. In *2016 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–6, Nov 2016.

[89] V. Vonásek, D. Oertel, S. Neumann, and H. Wörn. Failure recovery for modular robot movements without reassembling modules. In *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*, pages 136–141, Jul 2015.

[90] K. Wang, Z. Wang, and H. Zhang. Development of modular wall-climbing robot inspired by natural caterpillar. In *2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering*, volume 1, pages 293–297, Aug 2011.

[91] L. Wang, M. Xu, B. Liu, T. Jiang, S. Zhang, and J. Yang. Experimental study on morphology and kinematics of mudskipper in amphibious environments. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1095–1100, Dec 2013.

[92] M. Wang, M. Ceccarelli, and G. Carbone. Experimental experiences with a larm tripod leg mechanism. In *2014 IEEE/ASME 10th International Con-*

ference on Mechatronic and Embedded Systems and Applications (MESA), pages 1–6, Sep 2014.

[93] W. Wang, Q. Huang, and Y. Zhang. On the motor fault diagnosis based on wavelet transform and ann. In *the 31st Chinese Control Conference (CCC)*, pages 5339–5346, Jul 2012.

[94] T. E. Wei, G. M. Nelson, R. D. Quinn, H. Verma, and S. L. Garverick. Design of a 5-cm monopod hopping robot. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, pages 2828–2833 vol.3, Apr 2000.

[95] W. Wei and X. Dawei. A new design and analysis of compliant spine mechanism for caterpillar climbing robot. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 790–795, Dec 2012.

[96] K. Xu, F. Wu, and J. Zhao. Simplified online q-learning for lego ev3 robot. In *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, pages 77–80, Nov 2015.

[97] A. Yamaguchi, J. Takamatsu, and T. Ogasawara. Dcob: Action space for reinforcement learning of high dof robots. *Autonomous Robots*, 34(4):327–346, May 2013.

[98] C. Yang, K. Yuan, W. Merkt, T. Komura, S. Vijayakumar, and Z. Li. Learning whole-body motor skills for humanoids. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 270–276, Nov 2018.

[99] J.-M. Yang. Omnidirectional walking of legged robots with a failed leg. *Mathematical and Computer Modelling*, 47(11):1372 – 1388, 2008.

[100] J.-M. Yang, Y.-K. Park, and J.-G. Kim. Fault-tolerant gait planning of multi-legged robots. In J. Buchli, editor, *Mobile Robotics, Moving Intelligence*, chapter 28. IntechOpen, Rijeka, 2006.

[101] P. Yazdjerdi and N. Meskin. Actuator fault detection and isolation of differential drive mobile robots using multiple model algorithm. In *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 0439–0443, Apr 2017.

[102] J. Yong, W. Hongguang, F. Lijin, and Z. Mingyang. A combined logistic and model based approach for fault detection and identification in a climbing robot. In *2006 IEEE International Conference on Robotics and Biomimetics*, pages 1512–1516, Dec 2006.

[103] J. Yosinski, J. Clune, D. Hidalgo, S. Nguyen, J. C. Zagal, and H. Lipson. Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In *In Proceedings of the 20th European Conference on Artificial Life*, pages 890–897, 2011.

[104] H. Zhang, J. Gonzalez-Gomez, and J. Zhang. A new application of modular robots on analysis of caterpillar-like locomotion. In *2009 IEEE International Conference on Mechatronics*, pages 1–6, Apr 2009.

[105] G. Zhong, L. Chen, and H. Deng. A performance oriented novel design of hexapod robots. *IEEE/ASME Transactions on Mechatronics*, 22(3):1435–1443, Jun 2017.

[106] F. Zhou, B. Dong, Y. Li, K. Liu, and Y. Li. Module system monitoring and fault detection for modular and reconfigurable robot. In *2018 Eighth International Conference on Information Science and Technology (ICIST)*, pages 114–120, Jun 2018.

Appendix

A Abbreviations

The following abbreviations are used in this manuscript:

AFL	Acoustic-based Fault Diagnosis for Legged Robot
ANN	Artificial Neuron Network
CIQR	Caterpillar-inspired Quadruped Robot
DOFs	Degrees of Freedom
EA	Evolutionary Algorithm
EAD	Evolutionary Algorithm with Direction
FFT	Fast Fourier Transform
FLC	Fuzzy Logic Classifier
FR	Fuzzy Rule
GA	Genetic Algorithm
IMU	Inertial Measurement Unit
MDF	Medoid of Fuzzy Set
MF	Mean of Fuzzy Set
MUD	Mudskipper-inspired Movement
NCC	Normalized Cross-Correlation
ODE	Open Dynamic Engine
PLI	PSO-based Leg-loss Identification
PSO	Particle Swarm Optimization
SLMIC	Self-learning Mudskipper-inspired Crawling
SVM	Support Vector Machine
TGC	Trotting Gait Controller
VF	Voting of Fuzzy Set