Journal of the Korean Society of Marine Environment & Safety
Research Paper
Vol. 23, No. 5, pp. 574-584, August 31, 2017, ISSN 1229-3431(Print) / ISSN 2287-3341(Online) https://doi.org/10.7837/kosomes.2017.23.5.574

Optimizing Path Finding based on Dijkstra's Algorithm for a Quadruped Walking Robot TITAN-VIII

Van Tien-Nguyen* · Byong-Won Ahn**† · Cherl-O Bae***
* Graduate School, Mokpo National Maritime University, Mokpo 58628, Korea
** Division of Marine Engineering, Mokpo National Maritime University, Mokpo 58628, Korea
*** Division of Marine Engineering & Coast Guard, Mokpo National Maritime University, Mokpo 58628, Korea

4족보행 로봇 TITAN-VIII의 Dijkstra's Algorithm을 이용한 최적경로 탐색

Van Tien-Nguyen* · 안병원**† · 배철오***
* 목포해양대학교 대학원, ** 목포해양대학교 기관시스템공학부, *** 목포해양대학교 기관·해양경찰학부

Abstract : *In this paper, the optimizing path finding control method is studied for a Legged-robot. It's named TITAN-VIII. It has a lot of advantages over the wheeled robot in the ability to walk freely on an irregular ground. However, the moving speed on the ground of the Legged-robot is slower than the Wheeled-robot's. Consequently, the purpose of the method is presented in this paper to minimize its time when it walks to a goal. It find the path, our approach is based on an algorithm which is called Dijkstra's algorithm. In the rest of paper, the various posture of the robot is discussed to keep the robot always in the statically stable. Based on above works, the math formulas are presented to determine the joint angles of the robot. After that an algorithm is designed to find and keep robot on the desired trajectory. Experimental results of the proposed method are demonstrated in the last of paper.*

Key Words : *Quadruped Walking Robot, Wheeled Robot, Optimizing path finding, Dijkstra's algorithm, Joint angle*

요 약 : 본 논문에서는 보행로봇의 일종인 TITAN-VIII라 불리는 로봇을 이용하여 가장 짧은 경로를 탐색하여 이동하는 방법에 관한 연구를 나타낸다. 보행로봇의 경우 바퀴구동 로봇에 비해 불규칙한 지면 위를 자유로이 이동 가능한 장점 등을 가지고 있는데 반해 이동속도는 바퀴구동 로봇에 비해 느린 편이다. 따라서 본 논문에서는 목적지에 도달하기까지 시간을 최소화하는 최적경로 탐색 제어방법을 제시하였다. 경로를 탐색하기 위해 Dijkstra's algorithm라 불리는 알고리즘을 기반으로 하여 적용하였다. 또한 로봇이 항상 정적인 자세를 유지하는 로봇의 다양한 자세에 대해서도 다루었다. 로봇의 자세제어와 알고리즘을 통하여 로봇의 관절각 결정에 필요한 여러 수학방정식을 제시하였다. 그 후 원하는 궤적으로 로봇이 이동하고 탐색하는 알고리즘을 고안하였고, 제안한 방법의 결과를 실험으로 확인하였다.

핵심용어 : 4족보행 로봇, 바퀴구동 로봇, 최적경로 탐색, Dijkstra's algorithm, 관절각

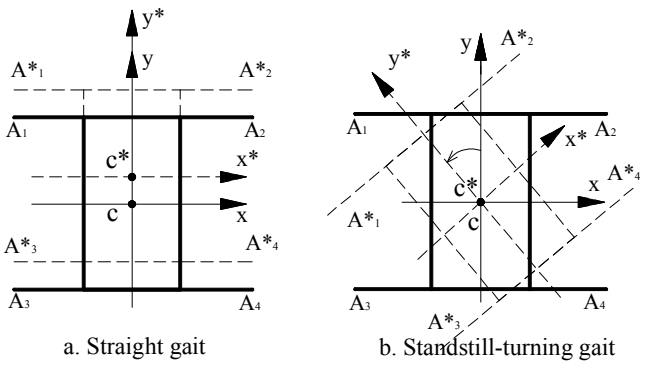
1. Introduction

Most the designed and developed of robots are to work in extreme environments such as ocean bottom, space and hazard prevention. It is not easy for the people living there because of the high atmosphere, the low temperature and the high risks for the life of people. However, the robots are capability of replacing people to reduce the risks. In fact, there are many different types of robots have been produced for many years, but they can be classified into two types according to the mobility on the ground. The first type is Fixed-robot. They don't have the ability of self-relocation and the second type of the robot is Mobile-robots. For Mobile-robots, they can freely travel by using the

* First Author : nguyenvantien@vimaru.edu.vn, 061-240-7232
† Corresponding Author : ds4cbt@mmu.ac.kr

self-locomotion ability. This paper only focuses on the last type that is Mobile-robots. Therefore, when the word "*robot*" that is mentioned in this paper, that mean is Mobile-robots. The self-locomotion of the robot can be performed in a lot of ways such as wheels, crawlers and fly. The robots are also separated into two types of Wheeled-robot and Legged-robot that depends on the mobility of the robot that is a wheel or a crawler, respectively.

The structure of a robot is mostly mechanical structure to form a kinematic chain of the robot. In addition, they also include their integration with sensors, actuator and controller system. For Wheeled-robot, it is easy to design a mechanism and a controller, but if the robot must work on the ground which the depression of a surface is greater than the diameter of wheel then the robot cannot pass over. The Legged-robot has a locomotion ability that is better than the Wheeled-robot's but it is very complex to design mechanical structure and the keep the stable status of Legged-robot in a moving situation. Because the kinematics of the legged robot is nonlinear, it means that the robot's parameter is varied according to the time (Chen et al., 2001). However, if the robot has to work on unfavorable environment like a rough terrain, then the best choosing in this situation will be the Legged-robot. The controller of the Legged-robot will be simpler by using a designed sequence of footholds (Fig. 1).



a. Straight gait b. Standstill-turning gait

Fig. 1. Describing the straight-going and standstill-turning gait of the Legged-robot.

The sequences are using as position set-points of the robot controller (Hirose et al., 2002; Hirose, 2001). In this case, the locomotion of the Legged-robot is only defined by two types of the moving. The first moving is called the straight-going gait and the other is called the standstill-turning gait (Chen et al., 2002; Izumi et al, 2001; Hirose, 1984). In the straight-going gait, the robot's direction is along the y-axis in the frame $\sum c$ (the $\sum c$ represents a frame which is attached with the robot's body) (Fig. 1a). In the standstill-turning gait, the robot will turn ϕ angle around the center of the robot's body, however, the position of the robot along y-axis which is not changed during the turning cycle time (Fig. 1b). The robot's center after finished a straight-going and standstill-turning gait which are denote by *c**.

If the position of the standstill-turning points is known then the robot is able to walk to anywhere by using straight-going and standstill-turning gaits sequence. In Fig. 2, it is shown an example of a method to go to a goal by using straight-going and standstill-turning gaits sequence.

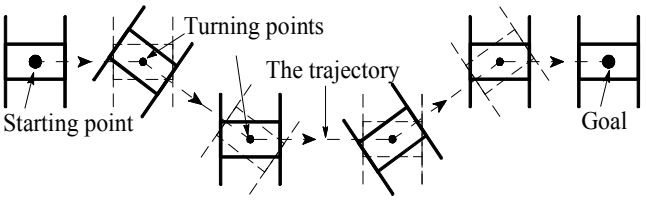


Fig. 2. The changing gait of legged robot.

Therefore, an algorithm has to develop to find these turning points for the robot. This issue have been mentioned in many researches (Chen et al., 2000; Pan and Cheng, 1991). Unfortunately, the researchers have not mentioned that how to minimize the robot's moving distance. In this paper, we proposed a new method to find the shortest way and avoid obstacles on the terrain by using Dijkstra's algorithm. In the robot's journey, it will meet not only small obstacles but also large obstacles.

For small obstacles, the robot can easily step over. Obviously, for large obstacles, the robot can't walk over the obstacles because the height of obstacles may be higher than the height of robot's footstep. In this situation, it is assumed that the obstacles on the terrain which are enough height to the robot can't step over or cross over, therefore, the best choice of the robot is walking around the obstacles. The Dijkstra's algorithm and turning point finding method are presented at Section 3. There are seven section in our research paper contain. Section 1 is introduction. Section 2 is the mechanical structure of quadruped robot TITAN-VIII. Section 4 discusses a way to obtain the generation of quadruped robot gait. Section 5 is simple flow chart of the control algorithm. The experimental results are shown in Section 6. Finally, Section 7 shows some of the conclusions drawn from this research.

2. Mechanical structure of TITAN-VIII

This paper is focused on TITAN-VIII that is a kind of the Legged-robot. A photo of quadruped walking robot TITAN-VIII is shown in Fig. 3. The major specification of TITAN-VIII in standard walking posture and is 0.3 [m/s] (craw gait) and 0.9 [m/s] (trot gait). The payload is 5~7 [kg]. Its leg length is 400mm and weight is 40kg. It is fed from battery or AC-DC power supply.

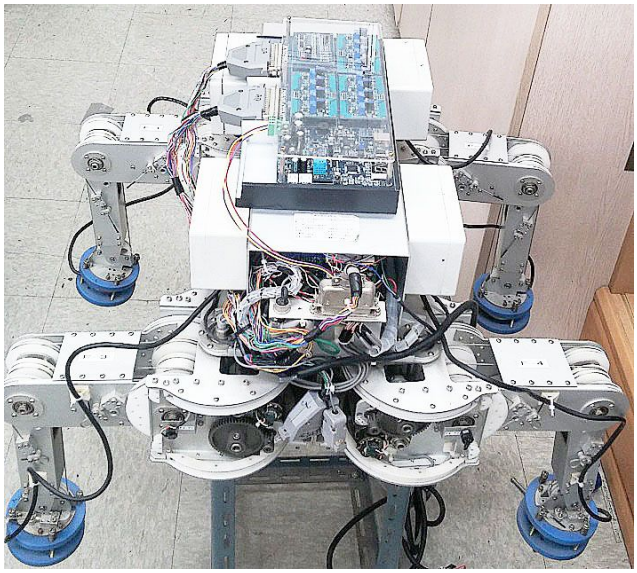


Fig. 3. The TITAN-VIII photo.

It is used to for general application purpose. The mechanism of the TITAN-VIII is a standard mechanism that composed of rotating joints. The robot's body is made of aluminum rectangular plate which is attached with four legs by rotating joints. These legs are driven by three actuators for each leg. The actuator consists of a DC motor and a motor driver. This will result in 12 actuators for 4 legs that make a more complex of the robot's construction, but it is making an ability to climb over a plane's slope.

The Fig. 4 is showed a block diagram of TITAN-VIII. It is included a Titech Robot Driver, a 16 channel D/A converter, a 16 channel A/D converter and a ARM Cortex-M3 microcontroller as a main controller. The desired joint angle of four legs can be automatically computed by main controller, after that it is transformed to the physical value by the D/A converter and sent to the motor driver. The real angles are on-line measured by means of the potentiometer to be sure that the joint angles reach to desired angles. The specifications of motor driver are U = 30 [V], Imax = 8 [A], Ucontrol = ±10 [V].

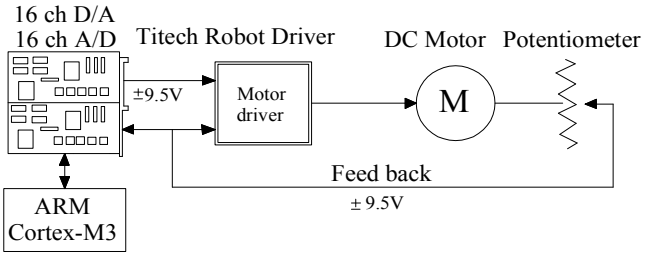


Fig. 4. Block diagram for one actual joint.

The schematic drawing of TITAN-VIII is presented in Fig. 5. In this figure, a, b, d and e are called the length of four link of four legs, respectively. The leg's mechanisms are composed of a link-wire plane mechanism, a rotating mechanism and a round plate as its foothold.

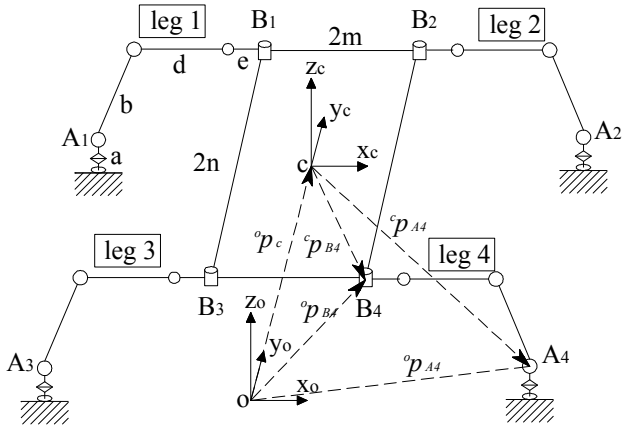


Fig. 5. Mechanical structure of TITAN-VIII.

In Fig. 5, the frame ∑c is denoted as a fixed frame at the robot's Center of Gravity (CoG) and another fixed frame is attached with the ground, is named ∑o. It is used to describe the robot's motion in the global frame. In the ∑*c*, the position vector of A_i and B_i are determined by ${}^{c}\rho_{A_i} = \left[{}^{c}x_{ai}, {}^{c}y_{ai}, {}^{c}z_{ai}\right]$ and ${}^{c}\rho_{B_i} = \left[{}^{c}x_{bi}, {}^{c}y_{bi}, {}^{c}z_{bi}\right]$ for *i = 1, 2, 3* and *4*, respectively. In addition, the correlation between ∑*c* and ∑*o* is expressed by vector ${}^{o}\rho_{c} = \left[{}^{o}x_{c}, {}^{o}y_{c}, {}^{o}z_{c}\right]$. A_i and B_i in ∑*o* are given by:

$$ {}^{o}\rho_{A_i} = {}^{o}\rho_{c} + {}^{o}T_{c}\,{}^{c}\rho_{A_i} \tag{1} $$

$$ {}^{o}\rho_{B_i} = {}^{o}\rho_{c} + {}^{o}T_{c}\,{}^{c}\rho_{B_i} \tag{2} $$

Where: *T* is the orientation matrix of ∑*c* with respect to ∑*o*

$$ {}^{o}T_{c} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3) $$

ϕ is a turning angle around center c, $\phi = 0$ means that there is no changing direction.

3. Dijkstra's algorithm

In most situations, the walking time of the robot to a desired point (a goal point) must be the shortest time for increasing of the robot's working efficiency. Therefore, the robot controller has to find the shortest way to the goal and avoid obstacles which appear on the terrain. There are a lot of methods to find the shortest way such as Dijkstra's algorithm, Bellman–Ford algorithm, Floyd–Warshall algorithm and Johnson's algorithm. The Dijkstra's algorithm was selected in this research because its operations are less complex than the others. It is very important in real-time control system of autonomous robot. In our experiment, we use a controller which is made by ourselves. Our robot controller has an ARM®Cortex®-M3 microcontroller and an AVR microcontroller without using a personal computer like the original robot controller. By using an independent controller, the robot can freely walk on working area without restricted by the connecting wires between the robot and the computer. In our controller, ARM®Cortex®-M3 has a higher processing performance than AVR microcontroller's, therefore, the path-finding algorithm is performed by ARM®Cortex®-M3 microcontroller and AVR for controlling peripheral devices such as actual motor drivers and sensors, respectively.

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph (Dijkstra, 1959) The algorithm was commonly used for two purposes. The first, it can be able to find the shortest path between a starting node and another node in a graph. The second, it can also be used for finding the shortest paths from a starting node to all other nodes. In the second purpose, for example, if the nodes are the cities and length between the pair of nodes represent driving distances then Dijkstra's algorithm can be used to find the shortest route way between one city and all other cities. Most situations, the robot only walks from a starting point to goal point, so the first method is a more accordant choice. Dijkstra's algorithm is discussed more detail at the following part.

The algorithm uses a graph as shown in Fig. 6a. The starting and goal point is called "s" and "d", respectively. In the graph, it is existed some the pair of vertexes which are called inter-vertexes, they have the ability to connect with other ones. The rest of ones do not have any this ability, they are seen as their distances between them are infinity.

Dijkstra's algorithm finds the shortest path between "s" and "d" in a graph as Fig. 6a. There are six vertexes in this graph numbering from 1 to 6. The distance is 15, 8 and 5 units from "s" to 2, 3 and 4, respectively. Vector V are defined to mount marked vertexes as "*a bag*" to contain intermediate vertexes in the operation of algorithm. The starting point is added into V like first element at the initialization stage. And now, the algorithm is starting at vertex 1 with only one element in V, V={1} and vertex 1 is called the current vertex. We can see that the pairs of 1-2, 1-3 and 1-4 are inter-vertexes and the shortest corresponding distance path between "s" and vertex 4 is 5 units. Therefore, vertex 4 will be marked a label as a current vertex and will be also added to vector V, then vector V has one more element and is expressed by V={1, 4}. So, vertex 4 is not connected to vertex 2 and vertex 5, the distance from vertex 4 to them is infinity. Obviously, the vertex 4 is connected to vertex 3 and vertex 6 but vertex 3 is selected as a candidate for adding V because the distance from vertex 3 to vertex 1 through current vertex 4 that is shorter than the distance from vertex 6 to vertex 1 through vertex 4, we have V={1, 4, 3}. The general formula for finding the length from a vertex to s that is presented as following.

$$ L_k(s,v) = \min\{L_{k-1}(s,v)\ ; L_{k-1}(s,u) + w(u,v)\} \quad (4) $$

Where $L_k(s, v)$ is a length from vertex v to starting vertex s at the cycle k and $L_{k-1}(s, v)$ is the length at before cycle of cycle k^{th}. $L_k(s, u)$ is the length from current vertex u to s and $w(u, v)$ is the length between vertex u and v. If there is more than one vertex with current vertex to make a pair of inter-vertexes then the vertex with the smallest L_k which will be selected as current vertex at next cycle. Additionally, we have to note that we don't need to consider vertexes which have been in V. In the Fig. 6, the current vertex is marked by symbol *.

And then, vertex 2 is chosen as a candidate for current vertex because the distance from vertex 2 to vertex s is equal to 10 units (from 2 to "s" through 3) and the distance is shorter than the distance from vertex 6 to vertex s. At this moment, V is added an element, V= {1, 4, 3, 2}. Because it is easy to see that the vertex 2 is only connected to vertex 5 , it becomes a next current vertex, V={1, 4, 3, 2, 5}.

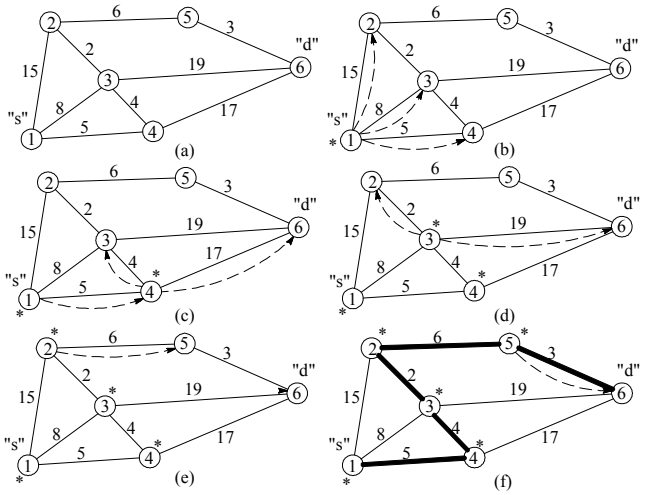


Fig. 6. Describing the graph of Dijkstra's algorithm.

In addition, vertex 5 is connected to the goal vertex as a unique connection. The algorithm will be stopped when the goal vertex is appeared in V vector. After all, we have V= {1, 4, 3, 2, 5, 6}. By finding vector V and saving the vertex tracings of current vertexes, the vertex are as turning-points of the robot which are obtained. Finally, we obtain the desired the robot path by connecting all the selected turning-points successively. The designed path is drawn in a solid line (Fig. 6f) which is generated from the starting point to the goal point. The algorithm is also used to determine a suitable path of the robot in an environment with many obstacles.

In Fig. 7, it is showing an example of the robot journey to destination. We figure out that the robot working environment is a plan with a gird. The meshes are made by vertical and horizontal lines with the same length between them. The lengths are a fixed value and are determined by an experimental equation. If the lengths are too small, the robot will not have enough space to perform self-standstill-turning gait. In addition, the robot path will be longer, if the lengths are more than the necessary value. The suitable length can be expressed by following experimental equation:

$$\triangle = 2n + S_{\max} \tag{5}$$

Where, $2n$ is the length of robot body. S_{max} is the longest footstep of the robot in a gait cycle and Δ is length of meshes. The purpose of making the gird is a technique to find suitable turning-points along the moving direction from the starting-point to the goal-point.

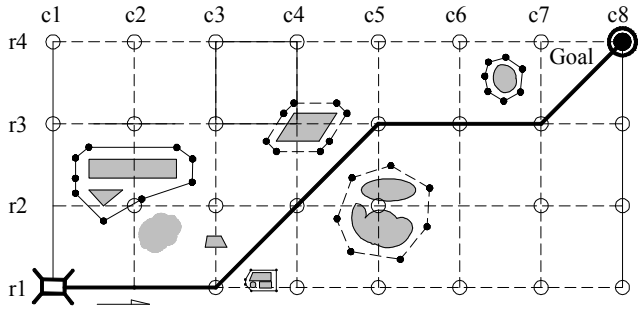


Fig. 7. The suitable path of the robot in an environment.

The crossing points between horizontal lines and vertical lines of the grid will be chosen as candidates for turning-point. At the crossing points, there are three ways to go to a next point that is along vertical, horizontal line and diagonal line as shown in Fig. 8.

We assume that the robot is being vertex 1, while the robot is able to select one of three directions. From current turning-point to vertex 2 denotes the direction vector from 1 to the goal, therefore, the vertex 2 is the next current position.

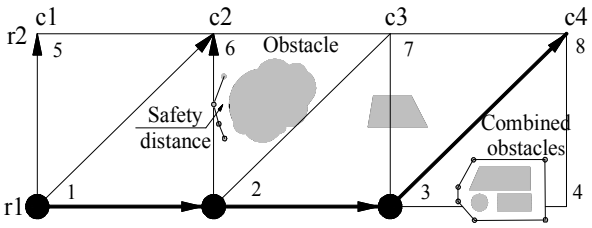


Fig. 8. Finding path to avoid the obstacles on terrain.

At the vertex 2, the diagonal line from 2 to 6 and 7 is prevented by an obstacle, therefore, the robot can only go straight to 3 along solid line. Obviously, when the robot is on 3, the diagonal line from 3 to 8 is selected such a unique line because others lines are inhibited by some obstacles on the environment. A boundary with the safety distance is imagined surrounding obstacles. The boundary will separate the working space of robot into two parts that are the forbidden and allowed zone. If an obstacle is too near others, the robot won't make a though way between them. In this situation, these obstacles are considered to be only one obstacle; they are called the combined obstacle. A specified line is generated surrounding the combined obstacles to make a region that prevent approaching robot to be nearer.

The obstacles on terrain will be detected by the robot's sensors such as the camera, proximity sensors and ultrasonic sensors. The

sizes of obstacles are measured by processing the acquired images from a camera to create a terrain map with fully obstacles.

4. Generation of the quadruped robot gait

In the middle of walking or running, a quadruped robot has to have balanced in order to avoid falling or unwanted body motion. Therefore, the controller has to generate a fit gait with the robot motion. There are many methods to generate a gait of Legged-robot as discussing in (Chen et al., 2001; Hirose et al., 2002). The purpose is to keep the projection of CoG onto horizontal plane within the supporting polygon which is created by supporting legs of the robot. In other words, the robot is being in the statically stable (Chen et al., 2001).

In the statically stable gait, the robot is able to remain stable as long as the CoG within the support area. It is shown an example of the quadruped robot in others status (Fig. 9). The black solid rounds are representing the robot foothold contacting ground which role such supporting legs on a working cycle of the robot and the white rounds are lifting-legs. In left part of Fig. 9, three legs provided a support area and the horizontal projection of CoG is located into the support area, therefore, the robot is in the statically stable and vice versa.

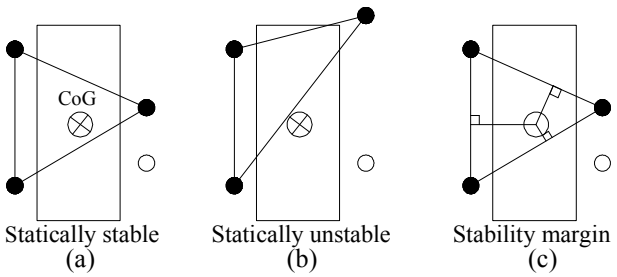


Fig. 9. Describing CoG and support polygon.

The shortest distance from CoG to the different edges of the support polygon is stability margin (Fig. 9). The stability margin provides some indication of the ability to resist disturbances while it is in the walking statically stable. The higher stability margin of the robot is the better because this is demonstrate that the robot is not easy to fall down when the robot is working at varied terrain.

We will define a sequence of the legs of the quadruped robot to keep CoG onto the support area at any time, is shown in (Fig. 10).

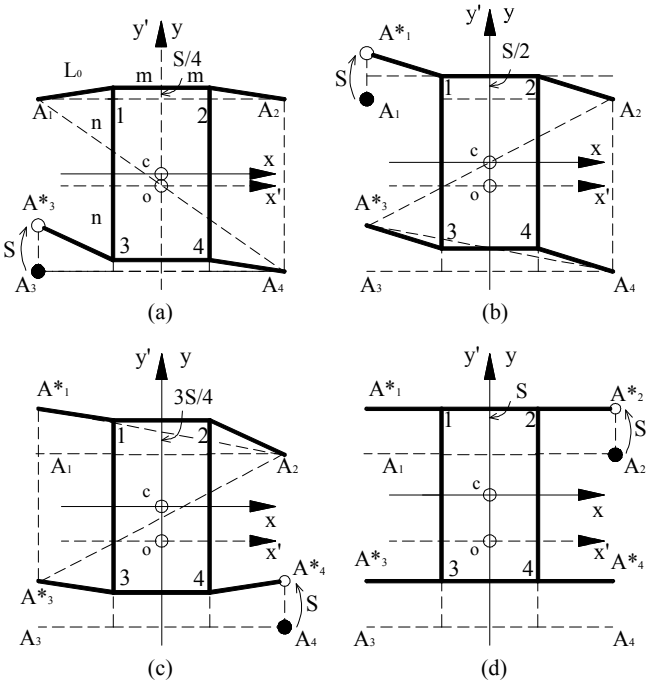


Fig. 10. The sequence of legs in straight-going gait.

For keeping CoG on to support polygon, three legs has to support its body at any time and one leg left that will play a role as the swing leg (lifting up). In a moving cycle, the support leg does a successive performance swing-ahead, although it does not lift up. With respect to the swing leg, it performs a sequence of the moving including three types of the moving that are lift-up moving, swing-ahead moving and put-down moving in a cycle. The robot repeats the sequences of the moving at the other cycles. The generation of quadruped robot gait to form the sequence of swing leg and support leg which is selected in such a way that keeping the vertical projection of CoG onto horizontal plane within the support area.

We define $\sum o$ and $\sum c$ like above. The quadruped robot is being in initial posture with the stride of legs and the height of the CoG that are denoted L_0 and H_0, respectively. Thus, the position vectors of four footholds in $\sum c$ are given by:

$$ {}^{c}\rho_{A_1} = [-L_0 - m, \quad n, \; h_1] \tag{6} $$

$$ {}^{c}\rho_{A_2} = [L_0 + m, \quad n, \; h_2] \tag{7} $$

$$ {}^{c}\rho_{A_3} = [-L_0 - m, \; -n, \; h_3] \tag{8} $$

$$ {}^{c}\rho_{A_4} = [L_0 + m, \quad -n, \; h_4] \tag{9} $$

And, the relation between $\sum o$ and $\sum c$ can be expressed by:

$$^{o}\rho_{c} = \left[^{o}x_{c},\ ^{o}y_{c},\ ^{o}z_{c}\right] \tag{10}$$

Where, h_i (i= 1, 2, 3 and 4) is height of the i^{th} foothold. Normally, the value of h_i is equal to zero but there is an exception that is when the robot climbs up and down on the stair. The height of the body is denoted by $^{o}z_{c}$. The robot body plate is always parallel with vertical plate and locates at a higher position than the ground; therefore, $^{o}z_{c}$ is not equal to zero.

When the robot begins to crawl forward, the third leg A_3 will be lifted up firstly after that it swings ahead a distance S (S is shorter than S_{max}) and put down at last of its duty, and finishes first step of a sequence of steps. The new position of A_3 is denoted by A^{*}_{3}. The foothold of other legs is still at the same position during this phase. It's clearly that the CoG is always settled in a supporting polygon which is made by A_1, A_2 and A_4. Obviously, the new position of the A_3 with respect to $\sum$c can be determined by following equations:

$$^{c}\rho_{A_3} = \left[-L_0 - m,\ -n + S,\ h_3\right]$$

The robot body is moved ahead a quarter of S after the first phase. The posture of the robot is varied as shown Fig. 9a. The position vector of CoG becomes:

$$^{o}\rho_{c} = \left[^{o}x_{c},\ ^{o}y_{c} + S/4,\ ^{o}z_{c}\right]$$

Similar to first selected swing leg, the others should be the next swing leg after the third leg. It is obvious that the robot body will be moved ahead a distance of S after completely four steps. The sequence of swing leg can be determined as 3→1→4→2 or 4→2 →3→1.

The sequence of legs in standstill-turning gait is presented as Fig. 11.

We assume that the turning angle is $\varnothing$. If the robot performs a left turning $\varnothing$ then the turning angle will be $\varnothing > 0$; otherwise, $\varnothing < 0$. Similar to the straight-going gait, the initial posture of the robot in standstill-turning gait is presented by solid line as shown in Fig. 11a. The A_4 leg can be selected as the first swing leg as shown in Fig. 11b. The A_2 leg should be the next swing leg. After A_2 and A_4 legs, the A_1 leg should be selected as the swing leg. Finally, the A_3 leg performs a footstep to position A^{*}_{3}, then the robot is returned to the initial posture. Therefore, the sequence of swing leg for left turning is 4→2→1→3; otherwise, the sequence is 3→1→2→4 for right standstill-turning gait about the robot center *c*. The foothold vector can be determined the same way with straight-going gait.

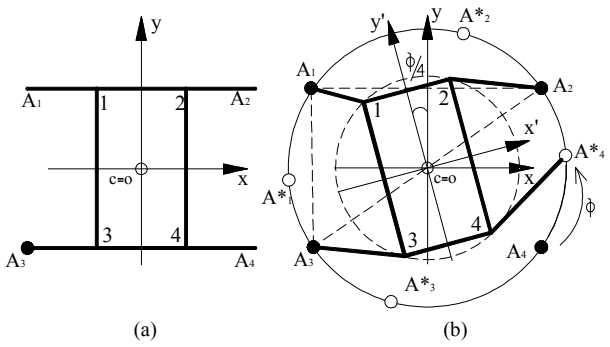


Fig. 11. The sequence of legs in standstill-turning gait.

TITAN-VIII has four legs and three activating joints in each leg (Fig. 12), where θ_{i1}, θ_{i2} and θ_{i3} are joint angles. Based on the generated desired gaits above, the position of footholds can be known at any time of a moving cycle. Now, the key for implementation of the desired moving is how to know angle joints of the robot. If we can know them, we can completely control the robot to follow a desired trajectory. It's easy to know that the value of θ_{i1}, θ_{i2} and θ_{i3} will be determined in relation of the height of the robot body H_i and the stride L_i, as shown in Fig. 12. When the robot moves, the formulations for determination of value of θ_i are generalized based on an equivalent leg mechanism of TITAN-VIII.

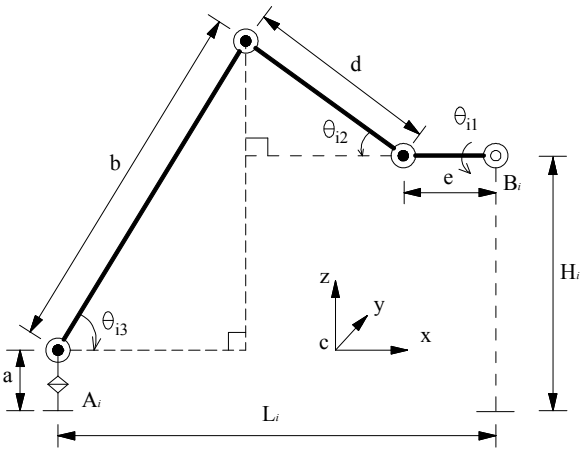


Fig. 12. Mechanism of TITAN-VIII leg.

Thus, we obtain equation:

$$\begin{cases} L_i = bcos\theta_{i3} + dcos\theta_{i2} + e \\ H_i = bsin\theta_{i3} + dsin\theta_{i2} + a \end{cases} \quad (11)$$

The other side, we can obtain the height of legs and stride from the position of foothold A_i and B_i in ∑c. We suppose that the shape of leg is constant from start to end of a moving cycle, and then we obtain:

$$\begin{cases} L_i = \sqrt{\left({}^c x_{ai} - {}^c x_{bi}\right)^2 + \left({}^c y_{ai} - {}^c y_{bi}\right)^2} \\ H_i = {}^c z_{ai} + {}^c z_{bi} \end{cases} \quad (12)$$

Where: ${}^c\rho_{A_i} = \left[{}^c x_{ai},\ {}^c y_{ai},\ {}^c z_{ai}\right]$ and ${}^c\rho_{B_i} = \left[{}^c x_{bi},\ {}^c y_{bi},\ {}^c z_{bi}\right]$, for *i = 1, 2, 3* and *4,* respectively. Moreover, the revolute joint of B_i is able to rotate around *z* axis, we have:

$$\tan\theta_{i1} = \frac{\left({}^c y_{ai} - {}^c y_{bi}\right)}{\left({}^c x_{ai} - {}^c x_{bi}\right)} \quad (13)$$

Therefore, we finally obtain (14) from (11), (12) and (13):

$$\begin{cases} bcos\theta_{i3} + dcos\theta_{i2} + e = \sqrt{\left({}^c x_{ai} - {}^c x_{bi}\right)^2 + \left({}^c y_{ai} - {}^c y_{bi}\right)^2} \\ bsin\theta_{i3} + dsin\theta_{i2} + a = {}^c z_{ai} - {}^c z_{bi} \\ \tan\theta_{i1} = \left({}^c y_{ai} - {}^c y_{bi}\right) / \left({}^c x_{ai} - {}^c x_{bi}\right) \end{cases} \quad (14)$$

We denote some constants as follows:

$$e_{ix} = {}^c x_{ai} - {}^c x_{bi};\ e_{iy} = {}^c y_{ai} - {}^c y_{bi};\ B_t = {}^c z_{ai} - {}^c z_{bi}$$
$$A_t = \sqrt{\left({}^c x_{ai} - {}^c x_{bi}\right)^2 + \left({}^c y_{ai} - {}^c y_{bi}\right)^2}$$
$$C_t = \left(A_t^2 + B_t^2 - b^2 - d^2\right);\ E_t = bC_t + d;\ F_t = bsinD_t$$

Solving equation (14), we can obtain the joint positions θ_{i1}, θ_{i2} and θ_{i3} as shown in equation (15).

$$\begin{cases} \theta_{i1} = atan(e_{iy}/e_{ix}) \\ \theta_{i2} = acos\left(B_t/\sqrt{E_t^2 + F_t^2}\right) + asin\left(E_t/\sqrt{E_t^2 + F_t^2}\right) \\ \theta_{i3} = acosC_t + acos\left(B_t/\sqrt{E_t^2 + F_t^2}\right) + asin\left(E_t/\sqrt{E_t^2 + F_t^2}\right) \end{cases} \quad (15)$$

Because of the limit of the robot's mechanism and the roughness of the ground profile. When the robot reach the maximum stretch of four legs, it takes a maximum stride. The maximum stretch, denoted by Amax, is defined by equation (16) (Chen et al., 2001).

$$A_{\max} = \sqrt{(b+d)^2 - \left(H_0 + \triangle h + \triangle \nu - a\right)^2 + e} \quad (16)$$

Where $\triangle\nu$ denote the roughness of the ground profile, H0 is the height of the robot center-of-gravity in the initial posture. The leg stretch of the quadruped robot should satisfy:

$$A \le A_{\max} \quad (17)$$

Equation (17) is boundary condition to solve equation (15).

The maximum stride and the maximum turning angle are denoted by S_{max} and $\varnothing_{max}$, respectively. The maximum stride is the longest footstep in a gait cycle. $\varnothing_{max}$ is a maximum turning angle of the robot's joints in a gait cycle when the robot takes a left turning or right turning. T, are mentioned (Chen et al., 2001).

5. The control algorithm

The flow chart of control algorithm is shown in Fig. 13. The parameters at initial posture, a robot staring point and a goal point are given as initial inputs for the algorithm. The ground profile of the terrain is determined by the sensor system and the vision system. The collected data from the system is used to create a grid map that consists of the information of obstacles. The grid map play a role Dijkstra's algorithm input data. Based on the information, the shortest trajectory to the goal point will be completely established by Dijkstra's algorithm. The robot position vector is known from the trajectory after that the joint angles are determined from equation (15). Finally, the quadruped robot is actuated though ROBOT DRIVER to DC motor. If the robot doesn't reach the goal point in a moving cycle then the control system will repeat the sequence of leg until the robot is on the goal.

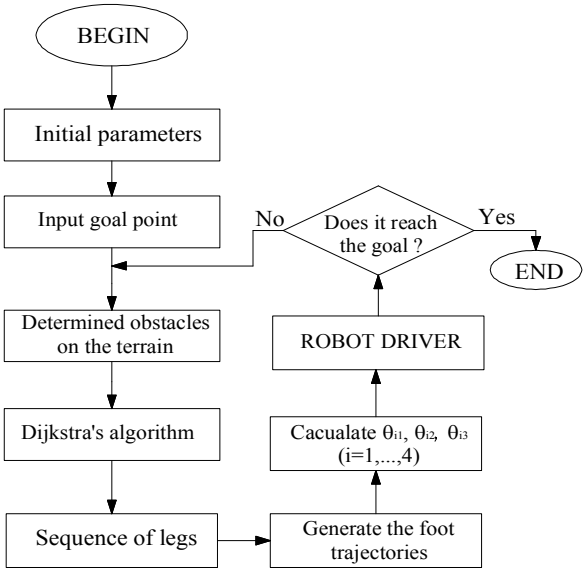


Fig. 13. The flow chart of control algorithm.

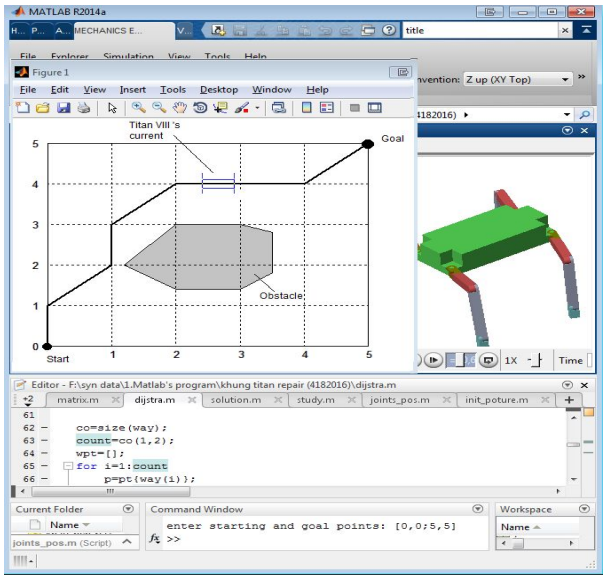


Fig. 14. TITAN-VIII simulation on Matlab & Simulink.

6. Experimental result

The robot in our experiment which is named TITAN-VIII, the size of robot body and robot legs are as follows [mm] m = 101, n = 201, a = 43, b = 200, d = 155 and e = 45. The initial posture is given by the position of legs with θ_{i1} = 0, θ_{i2} = 0 and θ_{i3} = 90^0, so that the length of legs and the height of body at the robot initial status are L_0= 200 mm and H_0= 243 mm, respectively. The largest stride and maximum turning angle in a gait cycle are S_{max} = 226 mm and $\emptyset_{max}$ = ± 44^0, (Hirose et al., 1984), respectively. If the desired turning angle is a larger turning angle than the robot maximum turning angle, the robot has to implement the stand still-turning gait for several times to reach the desired angle. The experimental results were conducted on a computer and there is simulation software installed on it. There are a lot of robot simulation software such as Matlab & Simulink, Gazebo, Webots and RoboDK. In our experiment, Matlab & Simulink was selected. The coordinates of a starting point and a goal point were given by user. A coordinate grid map was established on terrain and the robot was in a node of the grid. There is an obstacle on the terrain which the robot could step over; the robot had to find the shortest path to go to the goal. The simulation environment for the robot is presented in Fig. 14.

The bold line was the robot trajectory which was automatically created by using Dijkstra's algorithm. An obstacle was putted on the simulation environment that was indicated by the gray polygon (Fig. During the walking process on simulation environment, the robot avoided the obstacle by going around the obstacle because we assume that the robot can step over the obstacle. The left bottom corner is the starting point and right top corner is the goal point.

The change of variable joint θ_{i1}, θ_{i2} and θ_{i3} in a straight-going gait cycle were shown in Fig. 15.

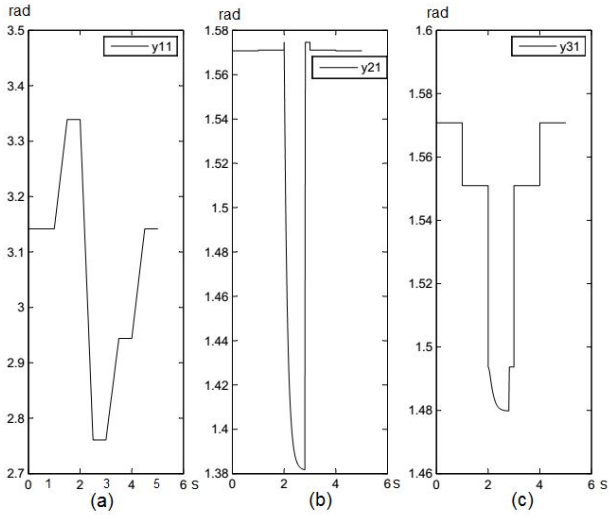


Fig. 15. The variable joints on a cycle.

A gait cycle is every 5s. At the time $t = 0 \div 1s$, when the robot was being in initial status, the variable joints were equal to initial value.

At time $t = 1 \div 2s$, the robot moved its leg 1st backward in the direction of the motion, when θ_{11} was increased at the first period and reached a stable angle at the end of period (Fig. 15a). It is easy to see that θ_{13} also was changed because the length of leg 1st was already increased a distance at this time.

At time $t = 2 \div 3s$, the leg 1st was as the swing leg, it was lifted from ground and was swung ahead. According to the sequence of steps in Fig. 10, the θ_{11} and θ_{21} were increased continuously to keep the leg's foothold 1st at the ground all the time $t = 3 \div 4s$ and $t = 4 \div 5s$ and were came back the initial angle at the end of cycle (Fig. 15b, c).

The change of the length of four legs in a moving cycle was shown in Fig. 16. The lengths of legs were changed between the initial status is L_0 = 200 mm and 215.7 mm.

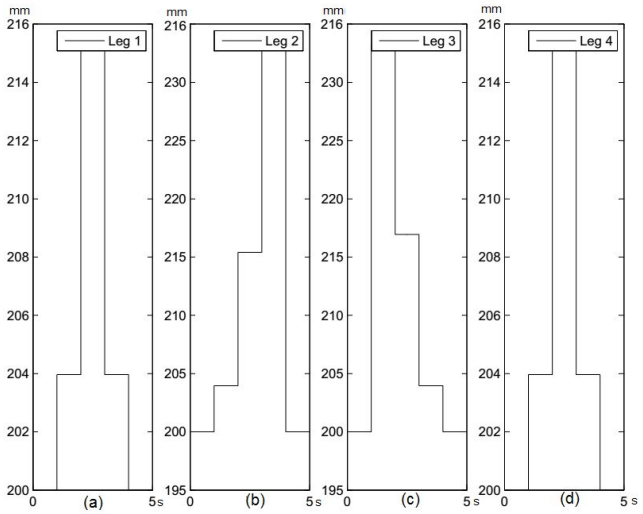


Fig. 16. The changing the length of four legs in straight-going gait.

Because the robot crawled without changing the height of CoG, the difference between H_i at first and last cycle is equal to zero. The height of four legs was presented in Fig. 17. At the turning point the robot will change its gait from the straight-going to the standstill-turning gait. The changing length of legs in this process were shown in Fig. 18. It is clearly that the robot turning angle is either $\pm 45^0$ or $\pm 90^0$, on the other hand, the maximum turning angle of the robot is $\emptyset_{max} = \pm 44^0$. The turning gait will be done two times when desired turning angle was $\emptyset = \pm 45$ and will be need four times when desired turning angle is $\emptyset = \pm 90$. After the standstill-turning, the robot walked toward in a new moving direction. The robot will repeat the sequence of moving until it reaches the goal.

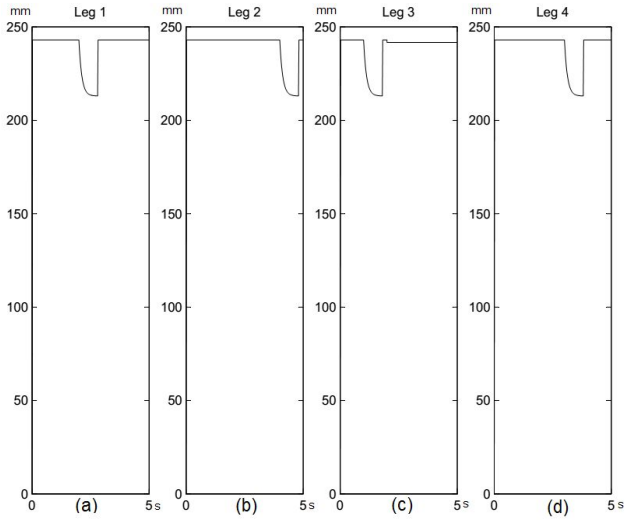


Fig. 17. The changing height of four legs in a cycle.

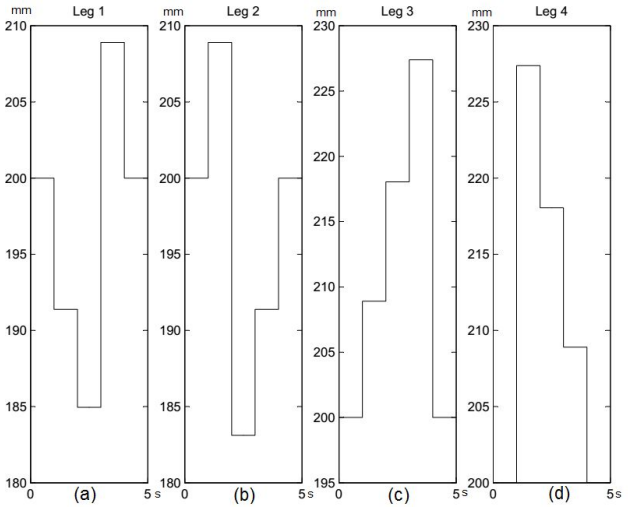


Fig. 18. The changing the length of four legs of the standstill-turning gait in a cycle $\emptyset = 45^0$.

7. Discussion

This paper has presented a method of generating a straight-going gait and a standstill-turning gait for a quadruped robot TITAN-VIII. The formulations of evaluating position of foothold in fixed frame have been mentioned in this paper to determine the joint positions in moving cycle of the robot. In

addition, the path finding algorithm has also been developed to avoid obstacles on the robot terrain. The obstacles avoidance algorithms based on Dijkstra's algorithm choose the suitable path to a goal. The proposed Dijkstra's algorithm is characterized by just finding the turning-point rather than the making a curve trajectory that reduces the calculation processing. It means that there is a fewer calculations processing than others, therefore, the sampling time is able to choose a small value to help increasing the robot speed.

The experimental results verified the reliability and effectiveness of the proposed method. It's proven by two things. First, the robot can perform a smooth transition from one gait cycle to a other one, because the robot returns to the initial posture after each gait cycle whenever it is in a straight gait or turning gait. Next, a possible large stride of the robot can be taken in very gait cycle by considering various factors in the robot status such as mechanism constraints, uneven terrain.

It should be also first research that the proposed method can be applied to general crawling with different duty factors. Moreover, the proposed method can be work in real world condition.

References

[1] Chen, X., K. Watanabe and K. Izumi(2000), Kinematic Solution of a Quadruped Walking Robot Posture Analysis of TITAN-VIII, Process of 14th IFAC World Congress, Vol. B, pp. 343-348.

[2] Chen, X., K. Watanabe, K. Kiguchi and K. Izumi(2001), Implementation of omnidirectional crawl for a quadruped robot, Advanced Robotics, Vol. 15, No. 2, pp. 169-190.

[3] Chen, X., K. Watanabe, K. Kiguchi and K. Izumi(2002), Path Tracking Based on Closed-Loop Control for a Quadruped Robot in a Cluttered Environment, Transactions of ASME, Vol. 24, pp. 272-280.

[4] Dijkstra, E. W.(1959), A note on two problems in connection with graphs, Numerische Mathematik, Vol. 1, No. 1, pp. 269-271.

[5] Hirose, S.(1984), A Study of Design and Control of a Quadruped Walking, Int. J. Robot. Res., Vol. 3, No. 2, pp. 113-133.

[6] Hirose, S., Y. Fukuda and H. Kikuchi(2002), The gait control system of quadruped walking vehicle, J. Robotics Soc, http://www.tandfonline.com/doi/abs/10.1163/156855386X00193.

[7] Izumi, K., K. Watanabe and R. Sato(2001), Behavior selection based navigation and obstacle avoidance approaching visual and ultrasonic sensory information for Quadruped robots, Advanced robotic systems, http://journals.sagepub.com/doi/full/10.5772/6234.

[8] Pan, J. and J. Cheng(1991), Study on quadruped walking robot climbing and walking down slope, Proc. IEEE / RSJ Int. Workshop on Intelligent Robots and Systems, Osaka, Japan, pp. 1531-1534.

Received : 2017. 05. 29.
Revised : 2017. 07. 13.
Accepted : 2017. 08. 28.