The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Linear–time Path and Motion Planning Algorithm for a Tree Climbing Robot – TreeBot

Tin Lun Lam, Guoqing Xu, Huihuan Qian and Yangsheng Xu

Abstract— **This paper proposes a path and motion planning algorithm for a tree climbing problem. This problem is challenging as the shape of tree is complex and irregular. To our best knowledge, this is the first paper dealing with the path planning problem on natural tree environment. Different from conventional motion planning approach that requires constructing a complex configuration space, this paper divides the planning problem into two parts, i.e., path and motion planning problem so as to reduce the dimension of the problem. An intuitive method to represent a climbing space is proposed that highly simplifies the path planning problem. With the use of a dynamic programming algorithm, an optimal path to reach a target position can be acquired in linear time. In addition, an efficient motion planning algorithm for a tree climbing robot named TreeBot is developed to make TreeBot follow the planned path.**

I. INTRODUCTION

Tree maintenance is important to protect both trees and human lives. To reach an upper part of trees for maintenance, workers always put a tool at the end of a long rod to help get close to the target position. However, it will become infeasible if the target position is very high. To solve this problem, the workers need to climb up the tree to perform the maintenance. Since tree climbing is a dangerous task for human, the development of a tree climbing robot is necessary to replace human work.

In literature review, there are only few robots designed for climbing on a tree. WOODY [1] is one of the tree climbing robots designed for replacing human works on tree. It climbs on trees by embracing a whole tree trunk. RiSE V2 [2] is a wall climbing robot that uses six legs to maneuver. It is demonstrated that the robot is able to climb up tree vertically. To our best knowledge, there is no related works for the motion planning problem on these tree climbing robots. The motion planning problem is challenging as the shape of tree is irregular and complex. There are many motion planning works for climbing on structured settings such as wall and glass window [3], [4], [5]. However, the natures of the structured settings are different from trees that make the motion planning methods not adoptable to the tree climbing problem.

In conventional motion planning approach [6], the configuration space (a set of possible transformations that could be applied to the robot [7]) of the problem must be constructed to help solve the problem. However, the formulation of the configuration space is complex as it involves complicated interaction between the environment and the robot kinematics. In addition, a high degrees of freedom (DOF) and continuous motion of a robot results in a high dimensional and huge configuration space that makes the problem difficult to solve.

T. Lam, G. Xu, H. Qian and Y. Xu are with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, and the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China. {tllam, gqxu, hhqian, ysxu}@mae.cuhk.edu.hk

Fig. 1. Prototype of the tree climbing robot - TreeBot

In this paper, an efficient motion planning algorithm for a tree climbing problem is proposed. The proposed algorithm is mainly designed for a novel tree climbing robot named TreeBot. The prototype of TreeBot is shown in Fig. 1. TreeBot is an innovative robot that is able to climb on many types of trees with high maneuverability. In order to solve the planning problem efficiently, the problem is divided into a path planning and a motion planning problem and solves it one by one so as to reduce the dimension of the problem space. In the path planning sub-problem, it is assumed that TreeBot is in point size and holonomic such that the kinematics of the robot can be ignored. It is aimed at finding an optimal path to reach the target position on the 2D manifold. The path planning algorithm includes several constrains to make the robot easy to follow. Since it only considers a 2D manifold of tree surface, the dimension of state space is relatively small.

An intuitive method is developed to represent the climbing space. It highly simplifies the path planning problem in linear-time complexity. A dynamic programming (DP) algorithm is adopted to find the optimal path according to

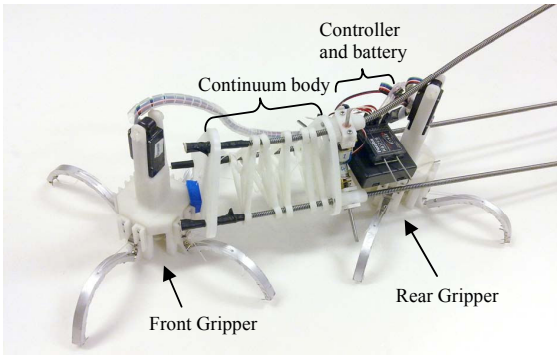

Fig. 2. Structure of TreeBot

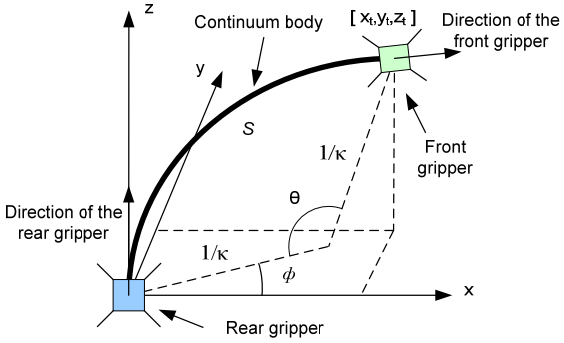

Fig. 3. Notations and configurations of TreeBot

specified constrains and requirements. As for the motion planning sub-problem, it is aimed to find an appropriate motion of the robot so as to follow the planned path. An effective strategy for motion planning is proposed that the solution can be obtained without any searching effort and state space formulation.

The following sections are organized as follows. Section II gives a brief introduction of the design and locomotion of TreeBot. The method of state space formulation is presented in Section III. The path planning algorithm is discussed in Section IV while the motion planning algorithm is presented in Section V. Experimental results are shown in Section VI. Finally, conclusions and future works are presented in Section VII.

II. INTRODUCTION OF TREEBOT

TreeBot is a tree climbing robot that composed of two grippers and a continuum body. The grippers are attached at the ends of the continuum body respectively. Fig. 2 illustrates the structure of TreeBot. The grippers are used to hold the robot on the tree surface while the continuum body is used for maneuver. The continuum body has three DOF which is able to bend and extend. The locomotion of TreeBot is similar to the locomotion of inchworm that moves forward by extending and contracting the body. The relationship among the robot body, front and rear gripper are illustrated in Fig. 3. The continuum body is in arc shape. According to [8], the shape of the continuum body can be represented by three parameters, i.e., S, κ and ϕ as shown in Fig. 3. Let the rear gripper is located at the origin, to put the front gripper to the target coordinate $\vec{P}_t = \begin{bmatrix} x_t & y_t & z_t \end{bmatrix}$, the posture of the continuum body becomes:

$$\phi = \tan^{-1}\frac{y_t}{x_t} \tag{1}$$

$$\kappa = \frac{2\cos\theta_1}{\sqrt{{x_t}'^2 + z_t^2}} \tag{2}$$

$$S = \theta/\kappa \tag{3}$$

where $\theta = 2\left(\frac{\pi}{2} - \theta_1\right)$, $\theta_1 = \tan^{-1}\frac{z_t}{{x_t}'}$ and ${x_t}' = x_t\cos\phi + y_t\sin\phi$.

III. STATE SPACE FORMULATION

Before working on the path planning problem, the state space to the problem should be formulated. A tree is composed of a trunk and numbers of branches. In this paper, a trunk is also treated as a branch. It is assumed that the relationship among branches can be represented by a tree data structure as illustrated in Fig. 4. To climb to a target position, there is a unique sequence of branches to go through. For example, if the target position is at branch 8 and the initial position is at branch 1, there is only one way to go, i.e., branch 1 $\rightarrow$ branch 4 $\rightarrow$ branch 8. The sequence can be obtained simply by using the backward search method in the tree data structure. As a result, for the path planning, the climbing space of other non-passing branches can be ignored. Nonetheless, the non-passing branches should be considered as obstacles.

To represent the climbing surface of each branch, the tree surface is discretized in numbers of points. The shape of tree is first decomposed by numbers of rings as shown in Fig. 5. The normal direction of a ring is equal to the growing direction of a shape of branch. The distance between each ring is separated in certain value such that the rings do not intersect. The shape of ring is defined by the outer shape of the specified position of a branch and hence it is not necessary be a perfect circle. Finally, each ring is equally discretized by certain numbers of points. Each point contains the information of the 3D Cartesian Coordinates and the normal vector of the surface of that point.

There are two situations that make TreeBot cannot reach a point: 1) the upper space of a position is not large enough for robot to go through which is a constant space for TreeBot. It may result by the upper space is occupied by other branches or 2) the gripping surface of a point is concave that the gripper cannot grip the surface tightly. The state space should contain the information of the unreachable points.

The information of the shape of tree can be obtained by many means such as laser or vision based sensing [9], [10]. Since this paper focuses on the planning problem, it is assumed that the shape of tree is given. The details of the sensing and the state space conversion problem will not be discussed in this paper.

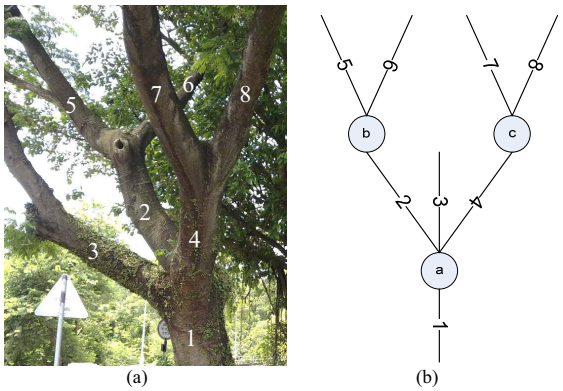

Fig. 4. Representation of the relationship among branches by tree data structure (a) Real tree structure; (b) Branches relationship represented in tree data structure.

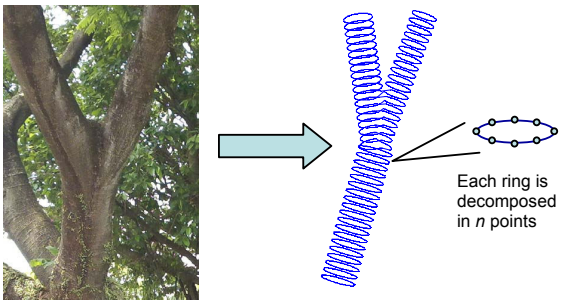

Fig. 5. Method of tree surface discretization

IV. PATH PLANNING

To go to the target position and avoiding obstacles are the basic requirements for path planning. In addition, to make a planned path easy to be followed by TreeBot, the path planning should have certain requirements. In order to eliminate the pull out force generated by the gravity, TreeBot should climb on a top side of the climbing surface as illustrated in Fig. 6 colored in red. Furthermore, the shorter path also reduces the energy consumption. It also implies a smoother path that makes TreeBot easier to follow. As a result, the path should be optimized so as to 1) go to the target position, 2) minimize the climbing distance, 3) climb on the top side of the climbing surface and at the same time 4) avoid obstacles.

A. Dynamic Programming

Dynamic programming (DP) is an efficient algorithm that is proved to be able to find global optimum solution to a problem [11]. It works well on discrete state that is difficult to search exhaustively. As a result, the DP algorithm is adopted to the path planning problem. To apply DP, the first step is to represent the problem in a DP formulation, i.e., to identify the *state*, *action*, *action value* and the *state value* to the problem.

State $S_{i,j}$: The states of the problem are the discrete points defined in Section III. The states are arranged in a matrix form with m rows and n columns that illustrated in Fig. 7. A state is denoted as $S_{i,j}$ where i and j denotes

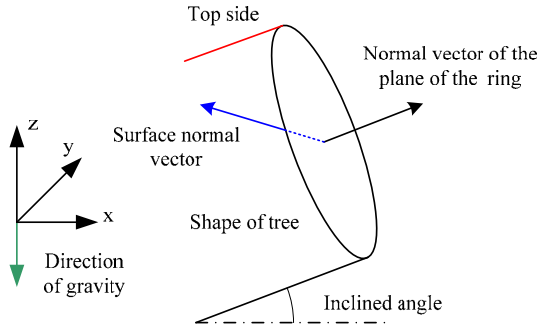

Fig. 6. Coordinates and notations of the shape of tree and the gravity vector.

the row and column in the workspace respectively. The first row represents the starting ring (ring 1) while the last row represents the ring that contains the target position (ring m). The elements in each row represent the points in that ring.

Action $S_{i,j} \rightarrow S_{i+1,k}$: It is assumed that the target position will not locate on the starting ring hence no repeated movement on a same ring is needed. The available movement is only the points on the next ring. This assumption is reasonable as the climbing motion rarely requires moving laterally without moving up or down. This assumption highly reduces the search space to the problem.

Action value $Q\left(S_{i,j}, S_{i+1,k}\right)$: The action value is defined as the sum of the reward values:

$$Q\left(S_{i,j}, S_{i+1,k}\right) = -D(S_{i,j}, S_{i+1,k}) + a_0 n_{i+1,k} + O_{i+1,k} \tag{4}$$

where $D(S_{i,j}, S_{i+1,k})$ represents the Euclidean distance between $S_{i,j}$ and $S_{i+1,k}$. $O_{i,j}$ is the obstacle value. The value is 0 if there is no obstacle and $-\infty$ if an obstacle presented. Obstacle means the unreachable point defined in Section III. a_0 is a positive scalar value to adjust the weight of $n_{i,j}$ in (4). $n_{i,j}$ relates to the amount of the pull out force generated by the gravity at that point. Refer to Fig. 6, the pull out force is directly proportional to the z component of the normalized surface normal vector $z_{i,j}$. Hence, the value of $n_{i,j}$ is defined as:

$$n_{i,j} = z_{i,j} - 1 \tag{5}$$

where $n_{i,j} \in [-2, 0]$.

State value $V_{i,j}$: By given a target position and the reward values, the state value of each state can be defined. The state value of row m-1 should solely be the distance to the target position, i.e.,

$$V_{m-1,j} = Q(S_{m-1,j}, S_{m,t}) \tag{6}$$

where $S_{m,t}$ denotes the target state.

The state value of other states can be found by:

$$V_{i,j} = \max\left(V_{i+1,k} + Q\left(S_{i,j}, S_{i+1,k}\right)\right) \tag{7}$$

where $k \in [1, n]$ and $i \in [1, m-1]$.

The possible next states of each state are the points in next row. As a result, by using DP, the computational complexity is only $O(mn^2)$. Actually, the value n is a problem

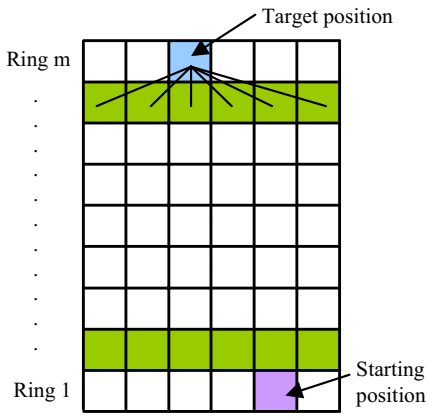

Fig. 7. State space representation for dynamic programming

independent value that will not change due to the height of tree. Therefore, the computational complexity should only be $O(m)$ that the problem can be solved in linear time.

Optimal Path: When the state value of each state is defined, the optimal path can be obtained by starting at an arbitrary position or the first row of state with maximum state value, then select the state in next row which the sum of state and action value $V_{i+1,k} + Q\left(S_{i,j}, S_{i+1,k}\right)$ is largest where $S_{i,j}$ and $S_{i+1,k}$ are the current and next state respectively.

B. Dynamic Environment

The environment of tree structure will rarely change in short time. The main reason of the change of environment may due to a more accurate information of the shape of tree is obtained when TreeBot gets close to that region. Since the calculation of state value is a top-down process, for a climb up motion, the change of environment at the lower part does not affect the state value at the upper part. As a result, only the state values at the lower part need to be modified. The path can then be updated according to the updated state values.

V. MOTION PLANNING

The path planning algorithm generates a 3D path on the manifold of tree surface with high likelihood of success. The next task comes to the motion planning to make TreeBot follow the planned path. The ideal solution is that all the steps of robot (front and rear gripper) and the robot body can place on the planned path. However, to find a motion that both front and rear gripper and the continuum body on the planned path may not feasible due to the nonholonomic constraints of the robot kinematics. It is assumed that there is a certain tolerance for the path following problem. This assumption is applicable as the path is planned to keep away from the obstacle to a certain distance. Searching methods can be applied to find the global optimal motion sequence to fit the planned path. However, it is time consuming. As a result, a computational efficient strategy to find out a suboptimal solution is developed instead of searching exhaustively.

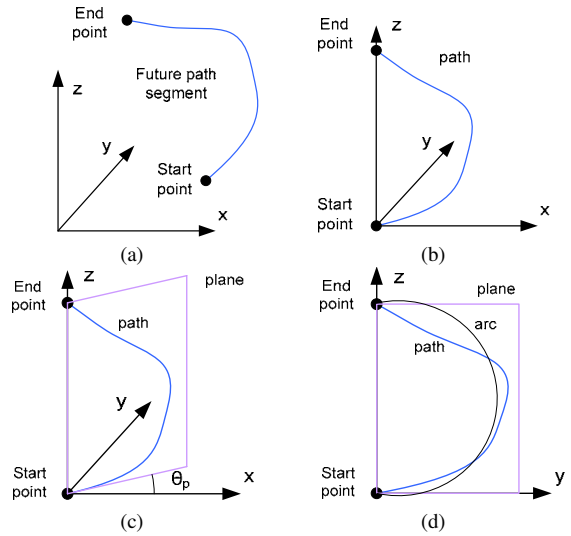

Fig. 8. Procedures for arc fitting: (a) Path segment; (b) Transformation; (c) Plane fitting; (d) Arc fitting

A. Motion Planning Scheme

Although it may not able to put both grippers and continuum body on the planned path, either one of the gripper can place on the path, then determines the position of another gripper so as to minimize the path following error. The front gripper based method, i.e., put all the steps of front gripper on the planned path, is adopted as it is much more intuitive. In this method, the extension motion is used to move the front gripper to the planned path while in the contraction motion, the rear gripper adjust the orientation of TreeBot to make the next extension motion best fit to the planned path. The procedures of the motion planning scheme are shown below.

1) Path Segmentation: As it is planned to place the front gripper on the planned path, the first task is to determine the target positions of the front gripper on the planned path. The path between the target positions of the front gripper are defined as path segments of the planned path. As TreeBot has variable length of step, the problem becomes how to determine the length of the continuum body in climbing motion. Since the gripping motion takes time, in order to climb up efficiently, the body in contraction motion should be as short as possible while the body in extension motion should be as long as possible so as to minimize the numbers of gripping motion. In order to simplify the problem, the length of contraction motion is set as the minimum admissible length while the distance between the target positions of the front gripper is set as constant. Once the planned path is segmented in numbers of pieces, the next task is to determine the optimal position of the rear gripper in each step according to the path segments.

2) Plane Fitting: Since the continuum body is in an arc shape, to find an optimal direction of the rear gripper that make the future motion fit the future path segment, the path segment should be approximated as an arc. To achieve this, the path segment is first fitted into a plane. It is assumed

that the start and end point of the path segment are on the plane. The path segment is first translated so that the start point is at the origin and then rotated (rotate θ_z about z-axis then rotate θ_y about y-axis) to put the end point on z-axis as shown in Fig. 8(b).

To find the fittest plane as illustrated in Fig. 8(c), an optimal θ_p should be determined such that the rotation about z-axis of θ_p results in the minimum absolute x component value. Let $[x_i, y_i, z_i]$ be the transformed coordinates of the path segment, θ_p can be obtained by minimizing:

$$x_i \cos\theta_p - y_i \sin\theta_p \tag{8}$$

By using the least square method and consider all the points:

$$\sum \frac{d}{d\theta_p}(x_i \cos\theta_p - y_i \sin\theta_p)^2 = 0 \tag{9}$$

Implies:

$$\tan\theta_p = \frac{-q \pm \sqrt{q^2+4}}{2} \tag{10}$$

where $q = \frac{\sum {y_i}^2 - \sum {x_i}^2}{\sum x_i y_i}$.

3) Arc Fitting: To find an arc to fit the path segment, the path segment is first transformed into 2D. Then the path is rotated about z-axis of θ_p so as to rotate the fitted plane to y-z plane. The path is projected on y-z plane thus becomes a 2D path as illustrated in Fig. 8(d). The approximated arc must pass through the start and end point of the path segment. Let the number of sampling points of the path segment be η, (y_1, z_1) and (y_η, z_η) be the start and end points of the path segment. (y_i, z_i) represents the points of the path where $i \in [1, \eta]$. Let (y_c, z_c) and r be the center and the radius of the approximated arc respectively. To cross the start and end point, the approximated arc should fulfill the following equations:

$$(y_1 - y_c)^2 + (z_1 - z_c)^2 = r^2 \tag{11}$$

$$(y_\eta - y_c)^2 + (z_\eta - z_c)^2 = r^2 \tag{12}$$

To combine (11) and (12):

$$z_c = a - b y_c \tag{13}$$

where $a = \frac{({y_1}^2 + {z_1}^2) - ({y_\eta}^2 + {z_\eta}^2)}{2(z_1 - z_\eta)}$ and $b = \frac{y_1 - y_\eta}{z_1 - z_\eta}$.

The distance error e_i of a point to the approximated arc can be found by:

$$e_i = (y_i - y_c)^2 + (z_i - z_c)^2 - r^2 \tag{14}$$

Sub. (11) into (14):

$$e_i = (y_i - y_c)^2 + (z_i - z_c)^2 - \left[(y_1 - y_c)^2 + (z_1 - z_c)^2\right]$$

$$\Rightarrow e_i = ({y_1}^2 + {z_1}^2) - ({y_i}^2 + {z_i}^2) - 2(y_1 + y_i) y_c - 2(z_1 + z_i) z_c \tag{15}$$

Sub. (13) into (15):

$$e_i = g_i + h_i y_c \tag{16}$$

where $g_i = ({y_1}^2 + {z_1}^2) - ({y_i}^2 + {z_i}^2) - 2a(z_1 + z_i)$ and $h_i = 2b(z_1 + z_i) - 2(y_1 + y_i)$.

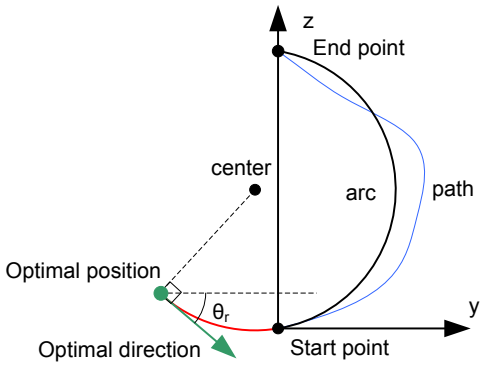

Fig. 9. Optimal position and direction of the rear gripper

By using the least square method and consider all data points, i.e.,

$$\sum \frac{d}{dy_c} {e_i}^2 = \sum \frac{d}{dy_c}(g_i + h_i y_c)^2 = 0$$

$$\Rightarrow y_c = -\frac{\sum g_i}{\sum h_i} \tag{17}$$

Once the value y_c is obtained, the value z_c and r can be found by (11) and (13) respectively.

4) Optimal Direction of the Rear Gripper: By arc fitting, the center and the radius of the approximated arc can be obtained. Then the optimal position and the direction of the rear gripper can be determined as illustrated in Fig. 9. In the figure, the green dot and arrow represent the optimal position and the direction of the rear gripper respectively while the red arc represents the contraction posture of the continuum body. The rear gripper can place at this position and direction only if the direction of the front gripper is tangential to the starting point of the approximated arc. However, the direction of the front gripper is uncontrollable when the position of the front gripper is fixed as it is a nonholonomic system. As a result, the optimal position is neglected and the target direction of the rear gripper is set to the optimal direction. By this method, the position of the rear gripper will shift out of the optimal position. The shift will not affect much of the path following result as the shift is small when compared to the length of path segment.

The optimal direction of the rear gripper $\vec{v}_{rg}$ in the global frame can be obtained by:

$$\vec{v}_{rg} = Rot_z(-\theta_z)\, Rot_y(-\theta_y)\, Rot_z(-\theta_p) \begin{bmatrix} 0 \\ \cos\theta_r \\ \sin\theta_r \end{bmatrix} \tag{18}$$

where $\theta_r = \tan^{-1}\frac{z_c}{y_c} - \left(\frac{\pi}{2} + \frac{S_{\min}}{r}\right)$, $S_{\min}$ is the length of the continuum body in contraction motion that colored in red in Fig. 9, $Rot_i(\theta)$ denotes the rotation matrix about i-axis in angle θ where $i \in x, y, z$.

B. Motion to Target Position

1) Front gripper: The target positions of the front gripper are defined by the path segmentation. To determine the posture of the continuum body to place the front gripper

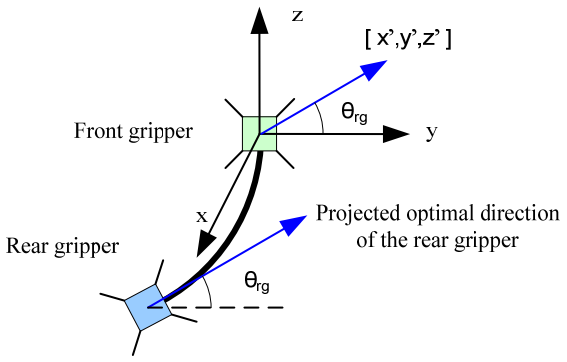

Fig. 10. The concept to determine the posture of continuum body in contraction motion

on the target position, the target position is first transformed into the rear gripper frame, i.e., the center of the rear gripper is at the origin and the direction vector of the rear gripper is on the z-axis as shown in Fig. 3. Then the posture of the continuum body to place the front gripper to the target position can be obtained by (1), (2) and (3).

2) Rear gripper: The optimal direction of the rear gripper is determined in Section V-A.4. To put the rear gripper in this direction, the position of the rear gripper may not on the tree surface. It is assumed that the surface at the target position of the rear gripper and the position of the front gripper has similar properties as their distance is short in the contraction motion. To make the target position of the rear gripper on the tree surface, the optimal direction vector is projected on the plane defined by the surface normal at the front gripper position. As a result, to find the posture of the continuum body to place the rear gripper, $\vec{v}_{rg}$ is first transformed to the front gripper frame that the center of the front gripper is at the origin, the direction of the front gripper is on the z-axis and the surface normal vector is on the x-axis as shown in Fig. 10. Then $\vec{v}_{rg}$ is projected on the y-z plane (arrow colored in blue in the figure). Finally, the posture of the continuum body to place the rear gripper can be determined as:

$$\phi = -sign\left(\theta_{rg}\right)\frac{\pi}{2} \tag{19}$$

$$\kappa = abs\left(\theta_{rg}\right)/S_{\min} \tag{20}$$

where $\tan\theta_{rg} = \frac{z'}{y'}$.

VI. EXPERIMENTS AND RESULTS

In order to evaluate the performance of the proposed path and motion planning algorithm, a tree model that composed of three branches is constructed. The tree surface is discretized as shown in Fig. 11. The rings are marked in different colors to distinguish the branches they belong to. The obstacles are marked as magenta.

A. Path Planning

To evaluate the path planning algorithm, the target position is located at the top of the branch 2 while the initial position is located at the bottom of branch 1 as shown in Fig. 11. Fig. 12 illustrates the reward value $n_{i,j}$ and $O_{i,j}$ of the

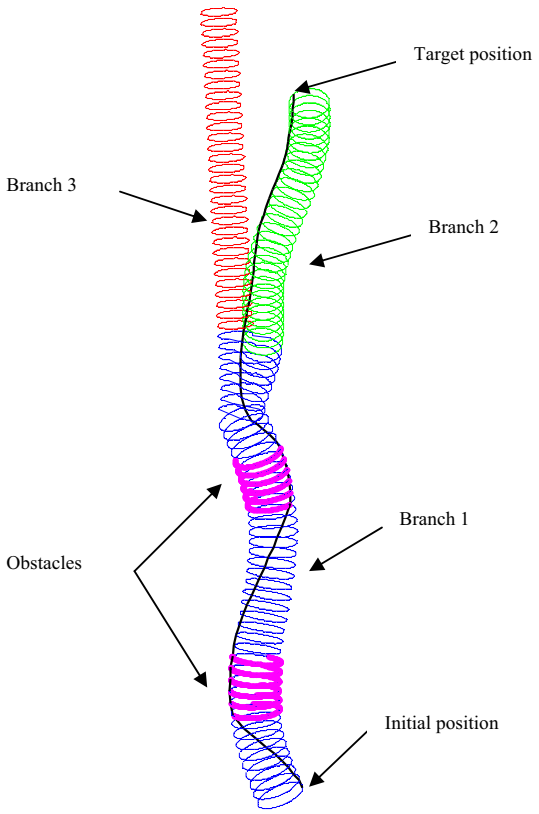

Fig. 11. Experimental tree model

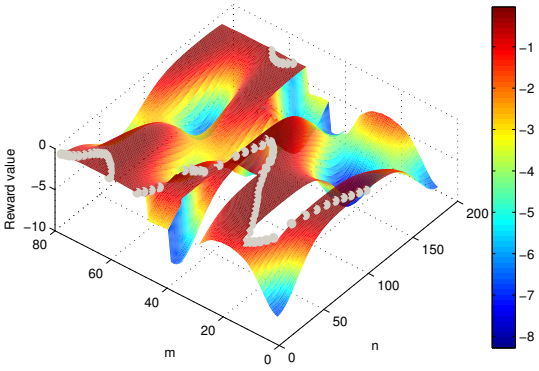

Fig. 12. Reward value of the selected state space

selected state space, i.e., branch 1 and branch 2. In the figure, the hollow regions represent the location of obstacles. The state space arrangement in Fig. 12 may not imply the actual geometric relationship. The planned path by the proposed path planning algorithm is shown in Fig. 11 colored in black and Fig. 12 colored in grey. In the figures, it can be observed that the planned path is succeeded to go to the target position by avoiding the obstacles and go through the positions with high reward value.

B. Motion Planning

Based on the planned path, the proposed motion planning algorithm has been applied. Fig. 13(b) illustrates the motions of the robot obtained by the proposed motion planning

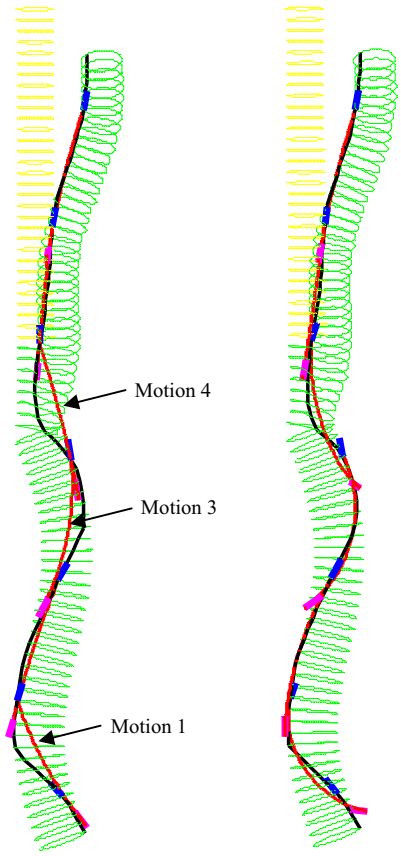

Fig. 13. Result of the motion planning. (a) Simple method; (b) Proposed method

algorithm. In order to show the motions of the robot clearly, the obstacles are not displayed in the figure. In the figure, the red line indicates the robot body while the blue and magenta arrows denotes the position and direction of the front and rear gripper respectively. The black line denotes the optimal path generated by the proposed path planning algorithm. It can be observed that the motions of the robot are close to the planned path. It shows the effectiveness of the proposed motion planning algorithm to track the planned path.

A simple motion planning method has also been applied for comparison that illustrated in Fig. 13(a). In the simple motion planning algorithm, the target direction of the rear gripper is simply equal to the direction of the front gripper. It can be noticed that by using the simple motion planning method, several motions (motion 1, 3 and 4) of the robot are far from the planned path. In that, the motion 1 and 3 went through the obstacles and hence resulted in fail climbing. The comparison of these two motion planning results reveals the significance of the proposed motion planning algorithm.

VII. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In conclusion, this paper proposed a global path and motion planning algorithm for a tree climbing problem that can be solved in linear time. An intuitive method is proposed to represent the climbing space so as to simplify the complexity of the problem. A dynamic programming algorithm is adopted to find the optimal climbing path that minimizes the climbing effort and avoids obstacles. A computational efficient motion planning algorithm for a tree climbing robot named TreeBot is also proposed to guide TreeBot to climb along the planned path. The proposed motion planning result is compared with a simple motion planning method. Result reveals that the significant improvement can be made by using the proposed motion planning method.

B. Future Works

The proposed motion planning algorithm segments the planned path in constant length. Although this method performs well in the experiment, an adaptive length of a path segment that according to the curvature of the future path may result in better performance. This method should be evaluated in the future. On top of that, a global optimal motion planning solution should be obtained in the future so as to compare the result with the solution generated by the proposed motion planning algorithm. Finally, the proposed algorithm will be implemented on TreeBot to evaluate the actual climbing performance in the future.

REFERENCES

[1] Y. Kushihashi, Y. Koji, et al. “Development of tree-climbing and pruning robot woody-1 - simplication of control using adjust function of grasping power” (in japanese). *Proceedings of JSME Conference on Robotics and Mechatronics*, pp. 1A1-E08, 2006.

[2] M. J. Spenko, G. C. Haynes, J. A. Saunders, M. R. Cutkosky, A. A. Rizzi, “Biologically Inspired Climbing with a Hexapedal Robot”, *Journal of Field Robotics*, vol. 25, no. 4-5, pp. 223-242, 2008.

[3] Z. Fu, Y. Zhao, Z. Qian and Q. Cao, “Wall-climbing Robot Path Planning for Testing Cylindrical Oilcan Weld Based on Voronoi Diagram”, *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 9 - 15, 2006, Beijing, China.

[4] C. Balaguer, A. Gimenez, J.M. Pastor, V.M. Padron and M. Abderrahim “A climbing autonomous robot for inspection applications in 3D complex environments”, *Robotica*, Vol 18, Issue 3, May 2000, Pages: 287 - 297, Cambridge University Press.

[5] Yeoreum Yoon and Daniela Rus, “Shady3D: A Robot that Climbs 3D Trusses”, *2007 IEEE International Conference on Robotics and Automation*, Roma, pp. 4071-4076, Italy, 10-14 April 2007.

[6] Steven M. LaValle, “Planning Algorithms”, *Cambridge University Press*, 2006.

[7] T. Lozano-Perez. “Spatial planning: A configuration space approach”, *IEEE Transactions on Computing*, C-32(2):108-120, 1983.

[8] Jones, B.A., Walker, I.D., “Kinematics for Multisection Continuum Robots”, *IEEE Transaction on Robotics*, vol. 22, no. 1, pp. 43- 55, February 2006.

[9] Dirk Hhnel, Wolfram Burgard, Sebastian Thrun, “Learning compact 3D models of indoor and outdoor environments with a mobile robot”, *Robotics and Autonomous Systems*, 44 (1), pp. 15-27, 2003.

[10] David Monnin, Armin L. Schneider, Frank Christnacher and Yves Lutz, “A 3D Outdoor Scene Scanner Based on a Night-Vision Range-Gated Active Imaging System”, *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pp. 938-945, 2006.

[11] Dreyfus, Stuart E., Law, Averill M., “The art and theory of dynamic programming”, *Academic Press*, 1977.