Motion Planning for Tree Climbing with Inchworm-like Robots

Tin Lun Lam and Yangsheng Xu
Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong
e-mail: tllam@mae.cuhk.edu.hk, ysxu@mae.cuhk.edu.hk

Received 19 September 2011; accepted 29 May 2012

This paper proposes a global path- and motion-planning algorithm that enables inchworm-like robots to navigate their way up tree branches. The intuitive climbing space representation method proposed here greatly simplifies the path-planning problem. The dynamic programming algorithm can be used to identify the optimal path leading to the target position in the target direction according to the constraints and requirements specified. The planned path can be applied in any tree-climbing robot that utilizes the nonenclosure gripping method. An efficient motion-planning algorithm for continuum inchworm-like robots is then developed to enable them to climb along the planned path with a high degree of accuracy. In comparison with the method proposed in our previous study, the method proposed herein significantly improves consistency between the planned path and the motions of the robot, and therefore makes it more practical to implement the motion-planning algorithm in trees of different shapes. The paper also describes hardware experiments in which the proposed planning algorithm is applied to enable inchworm-like robots to climb real trees, thus validating the proposed planning algorithm in practice. © 2012 Wiley Periodicals, Inc.

1. INTRODUCTION

Workers who are required to perform tasks on trees are often obliged to climb them. Given the dangerous nature of tree climbing, robots are expected to assist or replace humans in performing these tasks in the future. A literature review shows that several robots have been developed for tree climbing. One such climbing robot designed to replace human workers in the removal of tree branches is known as WOODY (Kushihashi et al., 2006). This robot climbs a tree by encircling its entire trunk with its arms and extending and contracting its body. Kawasaki et al. (2008) also developed a climbing robot for tree pruning that uses both a gripping mechanism inspired by lumberjacks and a wheel-based driving system for vertical climbing. Aracil et al. (2006) proposed a climbing robot that uses a Gough-Stewart platform to maneuver. It has greater maneuverability than the aforementioned two robots, and it can climb a branchless tree trunk with a certain range of bend. RiSE V2 (Spenko et al., 2008) is a wall-climbing robot that imitates the movements of insects in using six legs to maneuver. It has also been demonstrated that this robot is capable of climbing trees.

To the best of our knowledge, only one article deals with the motion-planning problem posed by tree-climbing robots (Aracil et al., 2006). However, the study merely discusses how the local motion-planning problem can be addressed according to local information. The global motion-planning problem has seldom, if ever, been discussed. In this context, the term “local” means that motion planning takes into account only the workspace near the robot, such as a segment of a tree. In contrast, the term “global” means that the entire workspace is considered, i.e., the entire shape of a tree. The global motion-planning problem is a challenging one as most trees have an irregular and complex shape. There are many global motion-planning approaches for climbing man-made structures such as walls (Fu et al., 2006), truss structures (Balaguer et al., 2000; Yoon and Rus, 2007), and floors (Jiang et al., 2009). However, because these man-made structures have regular shapes that differ from that of a tree, the approaches taken are not applicable to tree-climbing problems.

In the conventional motion-planning approach (LaValle, 2006), the configuration space [a set of possible transformations that could be applied to the robot (Lozano-Perez, 1983)] of a problem must be constructed to help solve the problem. However, formulating the configuration space is complex when it involves the nonholonomic constraints of robots, as these constraints restrict the geometry of feasible free paths between two configurations. A robot with high degrees of freedom and continuous motion requires a huge high-dimensional configuration space, which makes it difficult to deal with the planning problem. For example, Jiang et al. (2009) constructed the entire configuration space of the nonholonomic system and searched exhaustively for the optimal solution. Probabilistic sampling-based motion-planning methods such as rapidly exploring random tree (RRT) (LaValle, 1998) and probabilistic roadmap (PRM) (Kavraki et al., 1996) are some of the more efficient and popular approaches employed to
generate collision-free motion paths in the nonholonomic system. These methods have recently been used widely in footstep planning for legged robots (Perrin et al., 2011; Xia et al., 2009). However, the motion paths generated are not optimal with respect to different quality measures such as energy consumption and smoothness. An alternative method used to tackle the nonholonomic motion planning problem is to decompose the motion planning problem into two parts, as applied in Jiang et al. (1997), Sekhavat and Chyba (1999) and Aoude et al. (2007). In the first part, the optimal path connecting the initial position with the target position is found by ignoring the kinematic constraints of the system. In the second part, the motions required to enable the robot to follow the planned path are determined. This method can be used to obtain a suboptimal motion path according to specific quality measures.

Our previous study (Lam and Xu, 2011a) proposed a local motion-planning algorithm for tree climbing. This paper goes a step further in proposing an efficient global motion-planning algorithm for tree climbing. It is actually a motion-planning problem on a nonplanar two-dimensional manifold. Most of the motion-planning approaches described in the literature work in planar or three-dimensional space only, with few studies other than Papageorgiou et al. (2006) dealing with nonplanar two-dimensional manifolds. The algorithm proposed here can be applied to continuum inchworm-like robots such as Treebot (Lam and Xu, 2011b). As discussed in Lam and Xu (2011b), Treebot is capable of climbing a wide variety of trees with a high degree of maneuverability. To solve the planning problem more efficiently, it is decomposed into path-planning and motion-planning subproblems that are solved individually and orderly to reduce the dimensions of the problem space. In the path-planning subproblem, it is assumed that the robot is of point size and nonholonomic, such that its kinematics can be ignored. The path-planning algorithm includes several constraints and optimizations with respect to certain quality measures to make the robot reach the target position on a two-dimensional manifold easily. As it considers only the two-dimensional manifold of the tree surface, the dimensions of the climbing space are relatively small. An intuitive method developed to represent the climbing space greatly simplifies the path-planning subproblem in terms of linear-time complexity. A dynamic programming (DP) algorithm (Dreyfus and Law, 1977) is then utilized to find the optimal path according to the constraints and requirements specified. The planned path can be applied to any tree-climbing robot that utilizes the nonenclosure gripping method (in which the gripper does not encircle the gripping substrate). Analysis of the motion-planning subproblem is aimed at determining appropriate motions enabling continuum inchworm-like robots to climb along the planned path. An effective motion-planning strategy is proposed that allows for the solution to be obtained without additional state space formulation.

The work described in this paper is based on the work of Lam et al. (2010), with certain improvements to make it more practical to apply the planning algorithm in reality. The path-planning algorithm is improved to identify a smooth planned path to make it easier for a robot to follow. It is also capable of direction targeting to ensure that the last motion of the robot points it in the target direction. Furthermore, the motion-planning algorithm is improved by considering the fitness of the motion of the rear gripper. An adaptive path-segmentation method is also proposed, resulting in significant improvements in the degree of consistency between the planned path and the motion of the robot. Simulations conducted to illustrate the significant improvements brought about by the proposed motion-planning method are also reported. Hardware experiments are also conducted to demonstrate the performance of the proposed algorithm in reality.

The remainder of this paper is organized as follows. Section 2 briefly introduces the design and locomotion of Treebot. The climbing space formulation method is presented in Section 3. Section 4 discusses the path-planning algorithm. The motion-planning algorithm is detailed in Section 5. Section 6 reports and discusses the experimental results. Section 7 presents a conclusion and highlights possible directions for future work.

2. INTRODUCTION OF TREEBOT

Treebot is a tree-climbing robot composed of two grippers and a continuum body. Each of the grippers is attached to one end of the continuum body. Figure 1 illustrates the structure of Treebot. The grippers are used to hold the robot on the surface of the tree, with the continuum body being used for maneuvering. The continuum body has three degrees of freedom (DOFs) capable of bending and extending. The locomotion of Treebot is similar to that of an inchworm that moves forward by extending and contracting its body. Design and performance details can be found in Lam and Xu (2011b). Figure 2 presents a simplified model of Treebot, showing that its continuum body is in the shape of an arc. According to Jones and Walker (2006), the shape of the continuum body can be represented by three parameters, $S$, $\kappa$, and $\phi$, as shown in Figure 2. $S$, $\kappa$, and $\phi$ denote the length, curvature, and bending direction of the continuum body, respectively. Assuming the rear gripper is located at the origin, putting the front gripper in the target position $\hat{P}_f = [x_f, y_f, z_f]$ causes the posture of the continuum body to become

$$\begin{bmatrix} S_f \\ \kappa_f \\ \phi_f \end{bmatrix} = \begin{bmatrix} \frac{x_f^2 + y_f^2}{2x_f} \tan^{-1} \left( \frac{2k_f y_f}{x_f^2 - y_f^2} \right) \\ \frac{2k_f}{2x_f^2 + y_f^2 \cos^2 \phi} \tan^{-1} \left( \frac{y_f}{x_f} \right) \end{bmatrix},$$

where $\hat{x}_f = x_f \cos \phi + y_f \sin \phi$. 

Given the posture of the continuum body, the position of the front gripper can be obtained by

\[
\begin{bmatrix}
  x_f \\
  y_f \\
  z_f
\end{bmatrix} = \frac{1}{\kappa_f} \begin{bmatrix}
  \cos \phi_f \left[ 1 - \cos (\kappa_f S_f) \right] \\
  \sin \phi_f \left[ 1 - \cos (\kappa_f S_f) \right] \\
  \sin (\kappa_f S_f)
\end{bmatrix}.
\] (2)

3. CLIMBING SPACE FORMULATION

Before working on the path-planning subproblem, the climbing space must be formulated. A tree is composed of a trunk and branches. In the proposed algorithm, the trunk is also treated as a branch. It is assumed that the relationship among the branches can be represented by a tree data structure, as illustrated in Figure 3. To climb to a target position, a unique sequence of branches must be passed. For example, if the target position is at Branch 8 and the initial position is at Branch 1, then there is only one path to follow: Branch 1 → Branch 4 → Branch 8. This sequence can easily be obtained by using the backward search method in the tree data structure. It means that the climbing space of other nonclimbed branches can be neglected in the path-planning subproblem.

The tree surface is discretized by various numbers of points to represent the climbing surface of each branch. The shape of the tree is first decomposed into numbers of rings, as shown in Figure 4. The normal direction of a ring is equal to the growth direction of the shape of the branch. The distance between each ring takes a certain value such that the rings do not intersect. The shape of each ring is defined by the outer shape of the specified position of the branch, and thus it is not necessarily a perfect circle. Each ring is equally discretized by a certain number of points.

When a target position is given, the climbing space can be constructed by employing the discretized points belonging to the branches to be gone through. It can be
visualized in the form of a matrix with $m$ rows and $n$ columns, as shown in Figure 5. Each grid contains information on a discretized point, i.e., its three-dimensional (3D) Cartesian coordinates and the normal vector of the surface at that position.

There are two situations in which Treebot cannot reach a certain point. The first situation is when the upper space of a position is not sufficiently large for the robot to pass through, which may occur when the upper space is occupied by other branches. The second situation is when the gripping surface of a point is concave such that the gripper cannot grip the surface tightly. The climbing space contains information on such unreachable points.

Information on the shape of the tree can be obtained by several means, including laser- or vision-based sensing (Dreyfus and Law, 1977; Monnin et al., 2006). As this study focuses on the planning problem, the shape of the tree is assumed to be given and the details of the sensing and climbing space conversion problems are not discussed.

4. PATH PLANNING

The basic requirements of path planning are to devise a path enabling the robot to reach the target position and avoid obstacles en route. To enable a continuum inchworm-like robot to follow the planned path easily, the path should also fulfill certain additional requirements. To eliminate the pull-out force generated by gravity, the robot should climb on an upper apex of the climbing surface, as illustrated in Figure 6. Furthermore, a shorter path will reduce the robot’s energy consumption, and a smoother path will be easier for it to follow. As a result, the path should be optimized to minimize the pull-out force, climbing distance, and the angle of change.

4.1. Dynamic Programming

The dynamic programming (DP) algorithm is an efficient algorithm with a proven ability to find globally optimal solutions to a problem (Dreyfus and Law, 1977). Because it works well for discrete states that are difficult to search exhaustively, the DP algorithm is adopted for the path-planning problem. The first step in applying the DP algorithm is to represent the problem in a DP formulation, that is, to identify the state, action, action value, and state value of the problem.

State $S_{i,j}$: The states of the problem are the discrete points defined in Section 3. A state is denoted as $S_{i,j}$, where $i$ and $j$ denote the row and column in the workspace, respectively. The first row represents the starting ring (Ring 1) and 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 is not located on the starting ring, and thus no repeat movement will occur on a ring. Movement can only occur to the position on the next ring. This assumption is reasonable, as climbing motions rarely require lateral movement without moving up or down. This assumption significantly reduces the search space of the problem.

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

$$Q(S_{i,j}, S_{i+1,k}) = O_{i+1,k} + a_0 G_{i+1,k} - a_1 D(S_{i,j}, S_{i+1,k}) - a_2 \Delta (v_{i+1,k}, v(S_{i,j}, S_{i+1,k})),$$

(3)
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, which is taken as zero if there is no obstacle and as $-\infty$ if an obstacle is present. An obstacle means an unreachable position as defined in Section 3. $\Delta(a, b)$ is the angular difference between vectors $a$ and $b$, which can be obtained by the dot product. $v(S_0, S_0)$ is the direction vector from $S_0$ to $S_0$. $\hat{v}_{i+1,k}$ is the best direction vector at $S_{i+1,k}$. Including this term helps smooth the planned path and makes the final direction of the curve point in the target direction. $G_{i,j}$ relates to the amount of pull-out force generated by gravity at that position. The pull-out force is directly proportional to the $z$ component of the normalized surface normal vector $\hat{z}_{i,j}$, as shown in Figure 6. The value of $G_{i,j}$ is defined as

$$G_{i,j} = \frac{z_{i,j} - 1}{2},$$

where $G_{i,j} \in [-1, 0]$. $a_0$, $a_1$, and $a_2$ are the positive scalar values used to adjust the weight among each term.

To avoid the planned path being beyond the physical constraints of the robot, the nonlinear function can be applied in the action value such that

$$G_{i,j} = \begin{cases} \tan \left( \frac{G_{i,j} \pi}{G_{\text{min}} \pi} \right), & G_{i,j} \geq G_{\text{min}}, \\ -\infty, & G_{i,j} < G_{\text{min}}. \end{cases}$$

$$\Delta(a, b) = \begin{cases} \tan \left( \frac{\Delta(a, b) \pi}{\Delta_{\text{max}} \pi} \right), & \Delta(a, b) \leq \Delta_{\text{max}}, \\ -\infty, & \Delta(a, b) > \Delta_{\text{max}}. \end{cases}$$

where $G_{\text{min}}$ and $\Delta_{\text{max}}$ are the minimum and maximum constraints of $G_{i,j}$ and $\Delta(a, b)$, respectively.

State value $V_{i,j}$: Given target position $S_{m,t}$ and target direction $\hat{v}_{m,t}$, the state value of each state at the second last ring is defined as

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

where $\hat{v}_{m-1,j} = v(S_{m-1,j}, S_{m,t})$. The state values of other states can be found by

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

where $\hat{v}_{i,j} = v(S_{i,j}, S_{i+1,k}), k \in [1, n]$, and $i \in [1, m-2]$. $S_{i+1,k}$ is the state with the maximum value in Eq. (8).

Optimal Path: Once the state value of each state has been defined, the optimal path can be obtained by starting at a specific position or the state at the starting row with the maximum state value. The state in the next row for which the sum of the state and action values $V_{i,k} + Q(S_{i,j}, S_{i+1,k})$ is the largest is then selected, where $S_{i,j}$ and $S_{i+1,k}$ are the current and next states, respectively. The next possible states of each state are the states in the next row. As a result, using the DP algorithm means computational complexity is only $O(mn^2)$. In practice, the value $n$ is a problem-independent value that will not change with the height of the tree. Thus, the computational complexity required to solve the problem is only $O(m)$, which can be solved in linear time.

### 4.2. Dynamic Environment

The structure of a tree will rarely change within a short period. The climbing space has to be updated only when more accurate information on the shape of the tree is obtained once the robot moves closer to a particular region. State value calculations in the DP algorithm involve a top-down process. Therefore, for an ascending motion, a change in the environment in the lower part does not affect state values in the upper part, and only state values in the lower part need to be modified. The path can then be updated according to the new state values.

Another benefit of using the DP algorithm in this application is that once all the state values of the state space are obtained, it is easy to obtain in linear time the optimal path starting at an arbitrary position. As the robot may not reach the exact target position and orientation due to system error and disturbance, frequently updating the optimal path according to the current position of Treebot is necessary to moderate the path-following error.

### 5. Motion Planning

The path-planning algorithm generates a 3D path on the tree’s surface to connect the initial position with the target position. The next motion-planning task consists of ensuring that the robot follows the planned path. The ideal solution is for all the steps (of both front and rear grippers) and the body of the robot to be located on the planned path. However, finding a series of motions that keeps both the front and rear grippers and the continuum body on the planned path may not be feasible due to the nonholonomic constraints of the robot. It is assumed that the path-following problem has a certain degree of tolerance. This assumption is valid, as the path is planned to avoid obstacles by a certain distance. Although search methods can be used to find the globally optimal motion sequence that fits the planned path, they are time-consuming. It is therefore proposed that rather than conducting an exhaustive search, we should employ a computationally efficient strategy to find a near-optimal solution.

#### 5.1. Strategy of Motion Planning

It is difficult to maintain a situation whereby both grippers and the continuum body remain on the planned path. As an alternative, either one of the grippers can be placed on the path before the position of the other gripper is determined to minimize the path-following error. Due to its greater intuitiveness, we adopt a front-gripper-based method in which all of the steps taken by the front gripper are on the planned path. With this method, the extension motion is used to move the front gripper to the planned path and the contraction motion is used to adjust the rear gripper to ensure the next extension motion best fits the planned path. The procedure followed in the motion-planning scheme is discussed below.
5.1.1. Path Segmentation

As it is intended that the front gripper always remains on the planned path, the first task is to determine the target positions of the front gripper on the path. The paths between the target positions of the front gripper are defined as front path segments of the planned path. As the robot has a variable gait distance, the problem becomes one of determining the length of the continuum body in each climbing gait. A climbing gait is a cycle of locomotion including the body’s extension and contraction, as described by Lam and Xu (2011b). To climb efficiently, the body should contract and extend as much as possible to minimize the number of gripping motions, as they take time. The length of the contraction motion is thus set as the minimum admissible length $S_{\text{min}}$; the distance between the target positions of the front gripper is first set as the maximum length of extension of the robot body $S_{\text{max}}$. Once the planned path has been segmented, the next task is to approximate the segment in the shape of an arc.

5.1.2. 3D Arc Fitting

As the continuum body is in the shape of an arc, the path segment should be approximated as an arc to find the optimal direction of the rear gripper in which the future motion fits the future path segment. In addition, the target position of the rear gripper should be located near the planned path. As a result, unlike in Lam et al. (2010), the arc-fitting process also takes a rear path segment into account. The rear path segment is defined as the path below the current position of the front gripper with length $S_{\text{min}}$. The 3D arc fitting of each of the path segments can be derived in two steps: plane fitting and 2D arc fitting.

**Plane Fitting.** Let the start and end points of the front path segment be on the plane. The front and rear path segments are first translated so the start point of the front path segment is at the origin, and they are then rotated (rotated $\theta_z$ about the $z$-axis and then rotated $\theta_y$ about the $y$-axis) to locate the end point of the front path segment on the $z$-axis, as shown in Figure 7(b). In the figures, the solid and dashed lines represent the path segment and the rear path segment, respectively.

To find the plane with best fit as illustrated in Figure 7(c), the angle $\theta_p$ should be determined such that the absolute $x$ component values of the data are minimized by rotating $\theta_p$ about the $z$-axis. Letting $[x_i, y_i, z_i]$ be the transformed points of the path segments, where $i \in [1, \eta]$ and $\eta$ is the number of data points, $\theta_p$ can be obtained by minimizing

$$x_i \cos \theta_p - y_i \sin \theta_p.$$  

(9)

By using the least-squares method and taking all the points into account,

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

Figure 7. Procedures of arc fitting: (a) A path segment; (b) transformation; (c) plane fitting; (d) arc fitting.

Once the fitted plane has been obtained, the data are converted into two dimensions by projecting onto the plane and rotating $\theta_p$ about the $z$-axis onto the $y$-$z$ plane, as illustrated in Figure 7(d). The 2D arc-fitting process can then be applied.

**2D Arc Fitting.** The approximated arc must pass through the start and end points of the front path segment. Let $(y_1, z_1)$ and $(y_\eta, z_\eta)$ be the start and end points of the path segment, and let $(y_c, z_c)$ and $r$ be the center and the radius of the approximated arc, respectively. To ensure it crosses the start and end points, the approximated arc should satisfy the
following equations:

\[(y_1 - y_i)^2 + (z_1 - z_c)^2 = r^2, \quad (12)
\]

\[(y_\eta - y_i)^2 + (z_\eta - z_c)^2 = r^2. \quad (13)\]

Combining Eqs. (12) and (13),

\[z_c = a - by_c, \quad (14)\]

where \(a = (y^2 + z_c^2 - y_\eta^2 - z_\eta^2)\) and \(b = \frac{y_\eta - y_i}{z_\eta - z_i} = \frac{y_\eta - y_i}{z_\eta - z_c} = \frac{y_\eta - y_i}{z_\eta - z_c}.\)

The distance error \(e_i\) of a point to the approximated arc can be found by

\[e_i = (y_i - y_c)^2 + (z_i - z_c)^2 - r^2. \quad (15)\]

Substitute Eq. (12) into Eq. (15):

\[e_i = (y_i - y_c)^2 + (z_i - z_c)^2 - [(y_1 - y_c)^2 + (z_1 - z_c)^2] = (y_i^2 + z_i^2) - (y_1^2 + z_1^2) + 2(y_1 + y_i)y_c + 2(z_1 + z_i)z_c. \quad (16)\]

Substitute Eq. (14) into Eq. (16):

\[e_i = g_i + h_iy_c, \quad (16)\]

where \(g_i = (y_i^2 + z_i^2 - (y_1^2 + z_1^2) + 2a(z_1 - z_i))\) and \(h_i = 2[(y_1 - y_i) + b(z_1 - z_i)]\).

By using the least-squares method and considering all data points, i.e.,

\[
\frac{d}{dy_c} \sum e_i^2 = \frac{d}{dy_c} \sum (g_i + h_iy_c)^2 = 0, \quad (17)
\]

this implies

\[y_c = \frac{\sum g_i h_i}{\sum h_i^2} \quad (18)\]

Once the value \(y_c\) is obtained, the values \(z_c\) and \(r\) can be found from Eqs. (12) and (14), respectively.

The fitness of the fitted arc is measured by

\[e_{arc} = \sqrt{\frac{1}{\eta} \sum e_i^2} = \sqrt{\frac{1}{\eta} \sum \left(\frac{g_i + h_i \sum g_i h_i}{\sum h_i^2}\right)^2}. \quad (19)\]

5.1.3. Direction of the Rear Gripper

The center and radius of the approximated arc can be obtained by fitting the arc. The position and direction of the rear gripper can then be determined as illustrated in Figure 8. In the figure, the dot and arrow represent the desired position of rear gripper, respectively, which give the next motion the best fit to the path segment. The extended arc drawn from the fitted arc represents the contraction posture of the continuum body. The rear gripper can be placed in this position and direction only if the direction of the front gripper is tangent to the starting point of the approximated arc. However, given the nonholonomic nature of the system, the direction of the front gripper cannot be controlled when the position of the front gripper is fixed. As a result, the rear gripper follows the desired direction only and the position is neglected. With this method, the position of the rear gripper will shift marginally away from the desired position, although this does not have much of an effect on the path-following result.

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

\[
\vec{v}_{rg} = \text{Rot}_z(-\theta_r)\text{Rot}_y(-\theta_y)\text{Rot}_x(-\theta_x) \begin{bmatrix} 0 \\ \cos \theta_r \\ \sin \theta_r \end{bmatrix}, \quad (20)
\]

where \(\theta_r = \tan^{-1}\frac{z_\eta}{x_\eta} - \frac{\pi}{2} - \frac{\pi}{r}, S_{min}\) is the length of the continuum body in the contraction motion, and \(\text{Rot}_i(\theta)\) denotes the rotation matrix about the \(i\)-axis \((i \in x, y, z)\) in angle \(\theta\).

5.2. Posture of the Robot

5.2.1. Rear Gripper

If the rear gripper is pointing in the optimal direction, it may not be able to remain on the tree’s surface. It is assumed that the surfaces at the target positions of the rear gripper and the current position of the front gripper have similar properties, as the distance between them is short in the contraction motion. To ensure the target position of the rear gripper is on the tree’s surface, the optimal direction vector is projected onto the plane defined by the surface normal to the front gripper position. As a result, find an appropriate continuum body posture for placing the rear gripper, \(\vec{v}_{rg}\) is first transformed to the front gripper frame so 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 Figure 9. \(\vec{v}_{rg}\) is then projected onto the \(y-z\) plane (the dashed arrow in the figure). In the final step, the continuum body posture appropriate for placing the rear gripper can be determined.
5.2.2. Front Gripper

The target positions of the front gripper are defined by the path-segmentation process. To determine the appropriate posture of the continuum body to place the front gripper on a target position, the target position is first transformed into the rear gripper frame located at the target position of the rear gripper. The target position of the rear gripper can be obtained by Eq. (22). In the rear gripper frame, 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 Figure 2. The posture of the continuum body to place the front gripper at the target position can then be obtained by Eq. (1).

5.3. Adaptive Path Segmentation

Path segmentation at a constant length may induce two problems, namely the fitted arc being of poor quality and the target position of the front gripper being unreachable due to the body extension limit. These issues can be resolved by adaptively reducing the lengths of the path segments.

5.3.1. Quality of Arc Fitting

In some instances, path segmentation at a constant length may not result in a close fit to an arc as shown in Figure 7(d). A poorly fitted arc degrades the accuracy with which the path is followed, as the robot’s movement follows the shape of an arc. In this case, reducing the length of each path segment can help improve the fit. The length of the path segment decreases until the fitness value of the arc drops below a positive threshold, that is, $\max(e_{\text{plane}}, e_{\text{arc}}) < \epsilon$. This threshold cannot be too small or path segmentation may fail due to the possibility of the path not precisely fitting an arc shape.

5.3.2. Unreachable Target Position

In the motion-planning process, even the length of a path segment is defined as the admissible length of an extension of the robot’s body. The front gripper may not be able to reach the target position for two reasons: the first is that the approximated arc is longer than the admissible length of extension; the second is that the target position of the front gripper becomes more distant from the target position after the rear gripper moves. An unreachable target position for the front gripper can be detected by checking whether $S_f < S_{\text{max}}$ before the actual movement is undertaken. If this is the case, the length of the current path segment is reduced and the solution is recalculated until $S_f < S_{\text{max}}$.

Figure 10 shows the motion-planning procedure, which mainly consists of two loops. The outer loop divides the planned path into segments of a constant length, and it facilitates the motion planning and implementation process. The inner loop adjusts the length of the path segments to obtain a satisfactory solution.

6. EXPERIMENTS AND RESULTS

Evaluation of the proposed planning algorithm is divided into two parts. In the first part, simulations are conducted on a virtual tree. The results are compared with those of our previous method proposed in Lam et al. (2010). In the second part, hardware experiments are conducted on real trees to evaluate the actual performance of the proposed planning algorithm.

6.1. Simulation

As shown in Figure 11, a tree model with branches and obstacles is adopted in the simulations to evaluate the proposed path- and motion-planning algorithm. The obstacles in the figure are marked in black.

6.1.1. Path Planning

Figure 12 shows the path-planning results obtained with different target positions and directions. In the figure, the target directions are marked in cyan and the planned paths...
are colored black. The initial position is located at the bottom of Branch 1. It can be observed that the planned paths successfully reach the target positions and directions by avoiding the obstacles.

Figures 13 and 14 show the slope $\sin^{-1}(z_{i,j})$ and the angle of change $\Delta_1$ in the planned paths, respectively. The dashed line represents the limit of each physical constraint as noted in Section 4. The figures illustrate that the planned paths do not exceed these limits, thus demonstrating the validity of the planned paths.

In some cases, the planned paths exceed their physical constraints, meaning the target position and direction are unreachable with the given constraints. This is especially so when the target position is located beyond the limit. Any such occurrence indicates that another target position and direction should be selected.

6.1.2. Motion Planning

The simulation evaluating the motion-planning algorithm follows the planned path shown in Figure 12(c). The proposed motion-planning algorithm is compared with that presented in Lam et al. (2010). The parameters are set as $S_{\text{max}} = 100$ cm and $S_{\text{min}} = 15$ cm.

The planned path is divided into segments of a constant length using the method proposed in Lam et al. (2010). In doing so, only the front path segments are considered, with the rear path segments being ignored. Figure 15 shows the motion-planning results from the front and side views.
Figure 12. Path-planning results with different target positions and directions.

The black line in the figure is the planned path. The blue and red arrows indicate the direction and position of the front and rear grippers, respectively. The green arc represents the posture of the continuum body in the extension motion. The robot requires four climbing gaits to reach the destination. It can be observed that the path-following results are unsatisfactory, especially in the last two climbing gaits. Table I shows the length, plane-fitness values, and arc-fitness values of each path segment. It can be observed that the plane-fitness values or arc fitness values in the last two segments are large, indicating that the path segments do not closely fit an arc shape. The result, therefore, is significant path-following error.

The adaptive path-segmentation method is adopted by using the motion-planning algorithm proposed in this paper. As explained in Section 5, the arc-fitting method
Figure 14. Angles of change of selected states in different planned paths.

Table I. Length, plane-fitness values, and arc-fitness values of path segments obtained by using the motion-planning algorithm proposed in Lam et al. (2010).

<table>
<thead>
<tr>
<th>Path segment</th>
<th>Length (cm)</th>
<th>Plane-fitness value</th>
<th>Arc-fitness value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>94.9673</td>
<td>2.9603</td>
<td>0.9904</td>
</tr>
<tr>
<td>2</td>
<td>98.4937</td>
<td>0.4877</td>
<td>0.9272</td>
</tr>
<tr>
<td>3</td>
<td>97.8861</td>
<td>1.2775</td>
<td>0.3995</td>
</tr>
<tr>
<td>4</td>
<td>82.7126</td>
<td>2.2344</td>
<td>5.1185</td>
</tr>
</tbody>
</table>

considers both front and rear path segments, unlike the method set out in Lam et al. (2010). The plane- and arc-fitness values are restricted to below 0.6. Figure 16 illustrates the motion-planning results from the front and side views, and Table II lists the corresponding values. It can be seen that the path-following results are better than those derived from the algorithm proposed in Lam et al. (2010). There is no obvious path-following error. The proposed adaptive path segmentation technique results in five more gaits being required to reach the destination. Comparison of these motion-planning results reveals that the proposed motion-planning algorithm delivers a significant degree of improvement.

6.2. On Tree Experiments

To evaluate the proposed planning algorithm in real situations, hardware experiments are conducted on real trees. The first climbing target, the initial position, and the target position and direction are shown in Figure 17(a). The goal is for the robot to climb a trunk to a branch further up while negotiating an obstacle. The approximated tree model, the planned path, and the motions generated by the
Table II. Length, plane-fitness values, and arc-fitness values of the path segments obtained by using the proposed motion-planning algorithm.

<table>
<thead>
<tr>
<th>Path segment</th>
<th>Length (cm)</th>
<th>Plane-fitness value</th>
<th>Arc-fitness value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44.8199</td>
<td>0.4597</td>
<td>0.3869</td>
</tr>
<tr>
<td>2</td>
<td>40.4666</td>
<td>0.5265</td>
<td>0.4057</td>
</tr>
<tr>
<td>3</td>
<td>41.9786</td>
<td>0.1363</td>
<td>0.5303</td>
</tr>
<tr>
<td>4</td>
<td>65.8612</td>
<td>0.2740</td>
<td>0.4841</td>
</tr>
<tr>
<td>5</td>
<td>37.0491</td>
<td>0.4581</td>
<td>0.4040</td>
</tr>
<tr>
<td>6</td>
<td>62.3610</td>
<td>0.2815</td>
<td>0.5419</td>
</tr>
<tr>
<td>7</td>
<td>60.5054</td>
<td>0.5105</td>
<td>0.5167</td>
</tr>
<tr>
<td>8</td>
<td>9.5811</td>
<td>0.0478</td>
<td>0.5597</td>
</tr>
<tr>
<td>9</td>
<td>7.1478</td>
<td>0.5708</td>
<td>0.4440</td>
</tr>
</tbody>
</table>

Figure 16. Motion-planning results obtained by using the proposed motion-planning algorithm.

Figure 17. First experiment: (a) Climbing target, initial position, target position, and direction; (b) approximated tree model, planned path, and motions.

The proposed planning algorithm are illustrated in Figure 17(b). The length, plane-fitness values, and arc-fitness values of the corresponding path segments are listed in Table III. It takes around 30 s to compute the solution. The total length of the planned path is 141.4 cm. The motion-planning results show that the robot takes six climbing gaits to climb to the target position while avoiding the obstacle. It takes the robot around 145 s to reach the target position. Of the 11 climbing experiments conducted, one fails due to the front gripper not detaching smoothly in the third climbing gait. Table IV reports statistics on position errors and angular errors in comparison with the target position and direction in the ten successful trials, and some of Treebot’s climbing motions are shown in Figure 18. In these ten trials, Treebot successfully climbs near the target position and closely follows the target direction by performing the planned motions.

The second climbing target is shown in Figure 19(a). The goal is to climb up an obstacle-free trunk to a branch more distant than that employed in the first experiment. The approximated tree model, the planned path, and the motions generated by the proposed planning algorithm are illustrated in Figure 19(b). The length, plane-fitness values, and arc-fitness values of the corresponding path segments are listed in Table V. The total length of the planned path is 241.2 cm. It takes around 40 s to compute the solution. The motion-planning results show that the robot takes nine climbing gaits to climb to the target position. The target
Table III. First experiment: Length, plane-fitness values, and arc-fitness values of path segments obtained by using the proposed motion-planning algorithm.

<table>
<thead>
<tr>
<th>Path segment</th>
<th>Length (cm)</th>
<th>Plane-fitness value</th>
<th>Arc-fitness value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31.5895</td>
<td>0.1460</td>
<td>0.1380</td>
</tr>
<tr>
<td>2</td>
<td>29.4962</td>
<td>0.2350</td>
<td>0.2770</td>
</tr>
<tr>
<td>3</td>
<td>34.3770</td>
<td>0.2167</td>
<td>0.2970</td>
</tr>
<tr>
<td>4</td>
<td>21.2127</td>
<td>0.4580</td>
<td>0.6279</td>
</tr>
<tr>
<td>5</td>
<td>22.3665</td>
<td>0.3727</td>
<td>0.7686</td>
</tr>
<tr>
<td>6</td>
<td>5.6135</td>
<td>0.0050</td>
<td>0.0531</td>
</tr>
</tbody>
</table>

Table IV. First experiment: Statistics of the distance error and the angular error in ten trials.

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position error (cm)</td>
<td>2.61</td>
<td>1.09</td>
</tr>
<tr>
<td>Angular error (rad)</td>
<td>0.083</td>
<td>0.045</td>
</tr>
</tbody>
</table>

Table V. Second experiment: Length, plane-fitness values, and arc-fitness values of path segments obtained by using the proposed motion-planning algorithm.

<table>
<thead>
<tr>
<th>Path segment</th>
<th>Length (cm)</th>
<th>Plane-fitness value</th>
<th>Arc-fitness value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33.8212</td>
<td>0.4292</td>
<td>0.2565</td>
</tr>
<tr>
<td>2</td>
<td>11.8657</td>
<td>0.1173</td>
<td>0.5988</td>
</tr>
<tr>
<td>3</td>
<td>34.5561</td>
<td>0.2103</td>
<td>0.6589</td>
</tr>
<tr>
<td>4</td>
<td>33.2690</td>
<td>0.4758</td>
<td>0.2520</td>
</tr>
<tr>
<td>5</td>
<td>29.4552</td>
<td>0.5719</td>
<td>0.3346</td>
</tr>
<tr>
<td>6</td>
<td>31.1616</td>
<td>0.5631</td>
<td>0.4518</td>
</tr>
<tr>
<td>7</td>
<td>30.4311</td>
<td>0.2724</td>
<td>0.4035</td>
</tr>
<tr>
<td>8</td>
<td>30.5427</td>
<td>0.0632</td>
<td>0.1966</td>
</tr>
<tr>
<td>9</td>
<td>6.0552</td>
<td>0.0346</td>
<td>0.1751</td>
</tr>
</tbody>
</table>

Position is successfully reached in all ten of the climbing experiments conducted. Figure 20 shows some of Treebot’s climbing motions. It takes around 216 s for the robot to reach the destination. Table VI reports statistics on position error and angular error in comparison with the target position and direction in the ten trials. In these ten trials, Treebot successfully climbs near the target position and closely follows the target direction by performing the planned motions.

When comparing the results of these two experiments, it can be observed that the position and angular errors in the second experiment are larger than those in the first experiment. The main reason for this disparity is that the distance climbed in the second experiment is longer. The degree of error accumulates gait by gait due to many
uncertainties, such as slippage of the grippers, compliance of the continuum body, and imprecise modeling.

7. CONCLUSIONS AND FUTURE WORK

This paper proposes a global path- and motion-planning algorithm that can be used to resolve a tree-climbing problem. An intuitive method of representing the climbing space is proposed as an efficient means of reducing problem complexity. A dynamic programming algorithm is adopted to find the optimal climbing path that minimizes climbing effort and avoids obstacles. A computationally efficient motion-planning algorithm for continuum inchworm-like robots is also developed to ensure the robot climbs along the planned path. Simulations are performed to compare the proposed motion-planning algorithm with the algorithm put forward in Lam et al. (2010). The results reveal that the proposed method significantly improves the degree of consistency between the planned path and the motions of the robot. The findings, therefore, indicate that it is more practical to apply the motion-planning algorithm developed here to trees of different shapes. Hardware experiments in which the proposed planning algorithm is employed to enable Treebot to climb real trees are also conducted, demonstrating the practical value of the proposed planning algorithm.

The proposed method is not without its limitations. In the state described in this paper, Treebot follows the planned path in the open-loop control mode. This will cause path-following error to accumulate, and the experiments show that significant path-following error will be induced when the path is long. As a result, we are currently working on a method designed to track the actual position of the robot to update planned motions and eliminate cumulative error.

REFERENCES


high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12.
学霸图书馆

www.xuebalib.com

本文文献由“学霸图书馆-文献云下载”收集自网络，仅供学习交流使用。

学霸图书馆（www.xuebalib.com）是一个“整合众多图书馆数据库资源，提供一站式文献检索和下载服务”的24小时在线不限IP图书馆。

图书馆致力于便利、促进学习与科研，提供最强文献下载服务。

图书馆导航：

图书馆首页 文献云下载 图书馆入口 外文数据库大全 疑难文献辅助工具