VSP

Advanced Robotics 24 (2010) 943–962

ADVANCED ROBOTICS
brill.nl/ar

Full paper

Triple RRTs: An Effective Method for Path Planning in Narrow Passages

Wei Wang *, **Xin Xu**, **Yan Li**, **Jinze Song and Hangen He**

Institute of Automation, College of Mechatronics and Automation, National University of Defense Technology, Changsha, Hunan 410073, P. R. China

Received 18 March 2009; accepted 27 August 2009

Abstract

Although Rapidly-exploring Random Trees (RRTs) have been successfully applied in path planning of robots with many degrees of freedom under non-holonomic and differential constraints, rapidly identifying and passing through narrow passages in a robot's configuration space remains a challenge for RRTs-based planners. This paper presents a novel two-stage approach to address the problem of multi-d.o.f. robot path planning in high-dimensional configuration space with narrow corridors. The first stage introduces an efficient sampling algorithm called Bridge Test to find a global roadmap that identifies the critical region. The second stage presents two varieties of RRTs, called Triple-RRTs, to search for a local connection under the guidance of the global landmark. The two-stage strategy keeps a fine balance between global heuristics and local connection, resulting in high performance over the previous RRTs-based path planning methods. We have implemented the Triple-RRTs planners for both rigid and articulated robots in two- and three-dimensional environments. Experimental results demonstrate the effectiveness of the proposed method.

Keywords

Robot, path planning, Rapidly-exploring Random Trees, narrow passages, multi-d.o.f.

1. Introduction

During the past decades, path planning has been widely investigated in such areas as robotics, manufacturing, virtual prototyping, pharmaceutical drug design and computer animation [1, 2]. In high-dimensional configuration space, the dramatic decrease in efficiencies of the traditional deterministic path planning methods has led to the development of random sampling-based algorithms, which can be divided into two categories: multiple query and single query.

* To whom correspondence should be addressed. E-mail: wangmail200@yahoo.com.cn

 DOI:10.1163/016918610X496928

The probabilistic roadmap method (PRM) [3] is a typical multiple-queried method. It provides a powerful framework for path planning of multi-d.o.f. robots. The main idea of the PRM planner is to randomly sample a robot's configuration space and connect the sampled points to construct a roadmap graph that captures the connectivity of the collision-free configuration space. Owing to its simplicity and generalization, the original PRM planner was widely extended to the problems of path planning for closed chains [4–6], and multiple [7] and non-holonomic robots [8]. However, the uniform sampling strategy adopted by many PRM planners will reduce the efficiency dramatically when there are narrow passages in the free space. Great efforts have been devoted to address the problem by using a non-uniform strategy that samples more roadmaps near the boundary of the obstacles or in narrow passages to capture the connectivity of the critical regions reliably.

The PRM and its variants are suitable for multi-queried scenarios where randomly sampled roadmaps provide global landmarks for searching a feasible path. When they are applied to a one-shot planner, the performance still remains unsatisfactory as more computation time is spent in expensive preprocessing. Different approaches have been proposed for efficiently solving single-queried problems, including Ariadne's Clew [9], expansive space planner [10] and Rapidly-exploring Random Trees (RRTs) [11, 12]. The RRTs planner was initially proposed for robot path planning with non-holonomic and differential constraints. The algorithm constructs a tree-like data structure to incrementally explore unknown configuration space. Since there is no preprocessing in the framework of RRTs, the efficiency is usually high in the single-queried scenarios. In addition, the general RRTs algorithm is simple to implement. Therefore, it has attracted the attentions of the robotics community and found broad applications during the past decade. There are many variants of the RRTs algorithm to solve different path planning problems. For example, the RRT-Connect algorithm [13] combines a goal-oriented heuristics and bidirectional expansion strategy to manipulation planning of an articulated virtual human arm. Extensions of RRTs for closed articulated chains are introduced in Refs [14–16]. Adaptations of RRTs for kinodynamic and non-holonomic planning exist in [17, 18]. Dynamic-Domain RRTs is proposed to reduce the number of iterations by controlling the exploring orientation [19]. On the other hand, a deterministic, resolution complete alternative of RRTs is adopted in [20].

From the viewpoint of global path planning, RRTs planners can be treated as powerful tools for local path planning problems when a pair of configurations is input as a query. As there is not enough global information to accurately guide the process of tree expansion, they may spend lots of time in identifying and passing through unknown difficult regions. Li *et al.* [21] extend the original RRTs to continually explore unknown configuration space by using the data structure called Reconfigurable Random Forest (RRF) based on the historical experiences of previous multiple queries. Although the RRF algorithm can efficiently account for changeable environments by keeping a concise representation of pruned RRF, it is still unsuitable for a one-shot planner as the connectivity of queried roadmaps

cannot be verified in advance. Strandberg [22] generalizes the bidirectional RRT-Connect method directly to multiple trees in which several invisible samples are kept as the roots to spawn new trees. However, the algorithm cannot evaluate the importance of each sampled roadmaps, which may grow lots of useless trees to explore uninteresting local regions. Obviously, trying to connect the disconnected trees to the initial and goal trees is a time-consuming task that will not provide any benefit to find a feasible global path in a single-queried procedure.

In this paper, we present a novel two-stage approach to address the problem of multi-d.o.f. robot path planning in high-dimensional configuration space with narrow corridors. The first stage introduces an efficient sampling algorithm called Bridge Test to find a global roadmap that identifies the critical region. Next, the second stage presents two varieties of RRTs called Triple-RRTs to search for a local connection under the guidance of the global landmark. Except for two trees spawned from initial and goal configurations, respectively, the third tree is grown from the identified global roadmap in the Triple-RRTs framework, providing a global heuristics for RRTs explorations. The bidirectional connecting strategy is performed between a pair of selected trees so that each tree has equal opportunity to explore unknown regions and connect with the other tree. As a result, the two-stage strategy can take advantage of both global guidance of PRM and local connection of RRTs, resulting in high performance in finding a global collision-free path in a single-queried planner. Preliminary experiments have produced very encouraging results.

Section 2 reviews previous narrow passage sampling methods and then proposes an improved Bridge Test algorithm that can efficiently identify critical regions. Section 3 presents two varieties of Triple-RRTs methods based on the bidirectional RRT-Connect algorithm. Section 4 reports experimental results on rigid and articulated robots in two-dimensional (2-D) and 3-D environments. Finally, we draw our conclusions and prospect the future work.

2. First Stage: Narrow Passage Sampling

2.1. Related Work

The difficulty posed by narrow passages and their importance were first noted in early works of PRM planners, as their success highly depended on effective samples to capture the connectivity of free configuration space. Lots of non-uniform random sampling methods for PRM planners have been proposed to efficiently find global roadmaps to identify critical regions.

Hsu *et al*. [23] describe a technique based on a dilation of the free configuration space. The robot is allowed to slightly penetrate the obstacles and then free configurations are created in the neighborhoods of these penetrating configurations. Amato *et al*. [24, 25] present a number of techniques to sample along the boundary of obstacles, instead of wasting samples on large open areas that might be free of obstacles. Wilmarth *et al*. [26] propose a technique that samples near the medial

axis between a pair of obstacles, which is as far from the boundary of obstacles as possible. However, all of the above methods need geometric calculations that are expensive to implement in high-dimensional configuration spaces.

Is there any effective sampling strategy that does not explicitly depend on geometric shapes of the obstacles? The Gaussian sampler [27] is the first attempt to identify narrow passages without the need for geometric calculations. Instead, the algorithm adopts intersection tests in the robot workspace to obtain denser samples near the obstacle regions over the whole configuration space. However, in some cases, points near obstacle boundaries may lie far away from narrow passages due to its simple test on a pair of samples, more samples are wasted in uninteresting regions. Almost at the same time, a similar approach, called Bridge Test [28], has been proposed to identify narrow passages by using three samples along a line segment. Considering its high efficiency in finding globally important roadmaps to identify narrow passages, we improve the original Bridge Test algorithm as a primitive for our two-stage planner. In the next subsection, we will describe it in detail.

2.2. Bridge Test Algorithm

For clarity, we first give some definitions and terms. For a robot with n d.o.f., the n-dimensional topological space describing all possible positions and orientations of the robot is called the configuration space, denoted by $\mathcal{C}$. A configuration q is free if the rigid bodies of the robot placed at q do not collide with obstacles or with other bodies of the robot. The set of all free configurations in $\mathcal{C}$ is defined to be free space, denoted by $\mathcal{F}$. The obstacle space $\mathcal{B}$ is defined to be the complement of $\mathcal{F}$: $\mathcal{B} = \mathcal{C}/\mathcal{F}$. Thus, the path planning problem can be defined as follows. Given a pair of initial and goal configurations q_{init} and q_{goal}, find a continuous collision-free path $\tau : [0, 1] \to \mathcal{F}$, such that $\tau(0) = q_{\text{init}}$ and $\tau(1) = q_{\text{goal}}$.

A narrow passage is a small region that impacts the connectivity of $\mathcal{F}$. Uniform distribution sampling across the entire configuration space may not work well due to the small volumes of the narrow passages. As a result, denser sampling should be performed in the important local regions to capture the connectivity of the local geometries. The Bridge Test algorithm is designed to boost the sample density in narrow passages based on collision detection. In our implementation, only one sampled global roadmap is enough to identify a narrow passage as more samples in the critical region will be processed automatically in the successive procedure of Triple-RRTs exploration, which can reduce the computation time considerably.

Intuitively, a narrow passage in $\mathcal{C}$ has at least one restricted direction, along which small perturbations will cause collisions between a robot and obstacle. It is easy to sample at random a short line segment through a collision-free configuration q such that the two endpoints of this line segment lie in $\mathcal{B}$ (Fig. 1). The line segment resembles a bridge across the narrow passage so that it is called Bridge Test. If we successfully obtain such a line segment, we say that the point $q \in \mathcal{F}$ passes the test. Clearly building short bridges is much easier in narrow passage than in wide open free space.

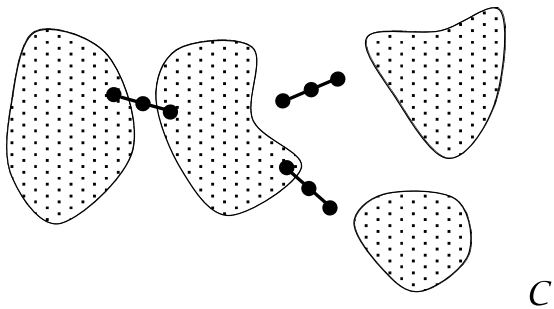

Figure 1. Illustration of Bridge Test. Note that only the point that lies in a narrow passage can pass Bridge Test.

In order to find a configuration that passes the Bridge Test, two endpoints must be selected in advance that should be assigned inside the obstacles and close enough to ensure the mid-point of the bridge lying in a narrow passage. The original Bridge Test algorithm samples the first bridge endpoint q_f from $\mathcal{C}$ randomly and then samples the second bridge endpoint q_s in the neighboring of q_f according to a specified probability density function λ_q, i.e., the product of the independent Gaussians on each axis of $\mathcal{C}$. The means of each Gaussians are set to be the position of q_f in $\mathcal{C}$. However, it is intractable to obtain the standard deviation σ for each independent Gaussian distribution, as σ depends on the width of the narrow passage that is always problem-specific. Instead, we directly use the intrinsic attribute in the sampling strategy to assign the second bridge endpoint q_s. Intuitively, the distance deviation between q_f and q_s in a narrow passage should not be too long. Uniform random sampling of q_s in $\mathcal{C}$ is not practical as q_s may lie far away from q_f. Therefore, we restrict the sampling region of q_s near q_f by the strategy described as follows. Suppose the lower boundary of the planning region in $\mathcal{C}$ is $q_{\min}$. A candidate configuration point q_c is sampled at random without checking for collision and then the value of $(q_c - q_{\min})$ is multiplied by a small scale $1/l$ to obtain the offset $dq = (q_c - q_{\min})/l$. q_s is then evaluated by:

$$q_s = q_f + (-1)^n \cdot dq, \tag{1}$$

where n is a random integer, implying that q_s may lie either in front of or behind q_f.

The improved Bridge Test (BRIDGE_TEST) algorithm is described in Algorithm 1. The algorithm either returns a mid-point of the bridge q_m that passes Bridge Test or returns failure.

2.3. *Influence of the Parameter* l

Compared to randomly sample configurations in $\mathcal{C}$ based on a multivariable Gaussian distribution, BRIDGE_TEST needs only one parameter l to scale the candidate configuration q_c into a bounded sampled region. We then analyze the influence of the parameter l so as to select the appropriate value for the setup of the experiments.

Algorithm 1
BRIDGE_TEST()

Step 1: The first bridge endpoint q_f is uniformly sampled at random in $\mathcal{C}$.

Step 2: If q_f is not collision-free (implying that q_f lies in obstacle space $\mathcal{B}$), the candidate configuration q_c is uniformly sampled at random in $\mathcal{C}$ as well. Otherwise, go back to Step 1.

Step 3: The second bridge endpoint q_s is calculated by: $q_s = q_f + (-1)^n \cdot (q_c - q_{min})/l$.

Step 4: If q_s is not collision-free (implying that q_s lies in $\mathcal{B}$), the mid-point of the bridge q_m is calculated by: $q_m = (q_f + q_s)/2$. Otherwise, go back to Step 1.

Step 5: If q_m is both collision-free and lies in $\mathcal{C}$ (implying that q_m belongs to $\mathcal{F}$), we say that q_m successfully passes Bridge Test. The algorithm returns q_m for the following procedure of Triple-RRTs planner and then terminates.

Step 6: Repeat Steps 1–5 until the maximum permitted number of iterations has been reached.

Suppose that the dimension of the configuration space is n. Defining the upper and lower boundaries of planning region in $\mathcal{C}$ to be $q_{max} = (u_1, u_2, \ldots, u_n)^T$ and $q_{min} = (r_1, r_2, \ldots, r_n)^T$, respectively. For each configuration $q = (q_1, q_2, \ldots, q_n)^T \in \mathcal{C}$, the inequality formula $r_i \leqslant q_i \leqslant u_i$ $(1 \leqslant i \leqslant n)$ must be held all the time.

The first and second bridge endpoints are defined as $q_f = (q_{f,1}, q_{f,2}, \ldots, q_{f,n})^T \in \mathcal{C}$, $q_s = (q_{s,1}, q_{s,2}, \ldots, q_{s,n})^T \in \mathcal{C}$, respectively. It is easy to derive that each element of q_s is bounded by:

$$q_{s,i} \in \left[q_{f,i} - \frac{u_i - r_i}{l}, q_{f,i} + \frac{u_i - r_i}{l} \right], \quad i = 1, \ldots, n. \tag{2}$$

Thus the maximum permitted width of the bridge W_{max} on the ith axis is evaluated by:

$$W_{max,i} = 2(u_i - r_i)/l. \tag{3}$$

If the minimum distance between two nearby obstacles is lager than W_{max} in an arbitrary direction, we say that no narrow passage exists in this corridor.

The illustrations of the maximum permitted region for both original (left two figures) and improved (right two figures) Bridge Test algorithms are shown in Fig. 2. The original Bridge Test algorithm introduces a Gaussian probability density function with standard deviation σ to restrict the sampling volume. The closer to the first bridge endpoint q_f, the higher probability the second bridge endpoint q_s is chosen. Thus, a shorter bridge is preferable in the original algorithm, which might cost a lot of computation time to search over the large configuration space. In our improved algorithm, uniform random sampling is performed to find q_s in a restricted region, where a narrow passage of which the length is smaller than the maximum permitted value could be identified with a predefined uniform probability distribution. Besides, more computation time is saved in our method since only one sampled landmark is enough to identify one narrow passage, and the neighboring difficult region will be explored by the successive RRTs expansion.

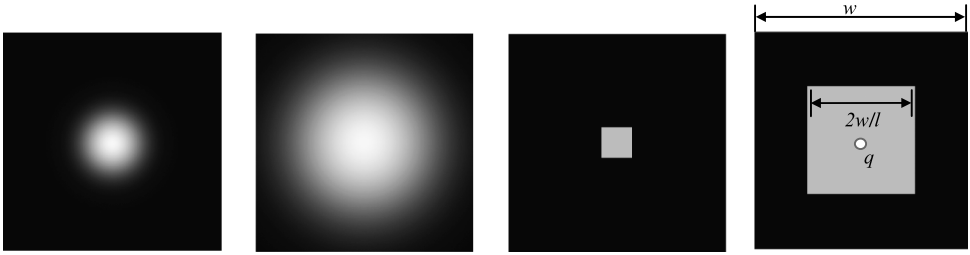

Figure 2. Sampling probability density of point q_s in the Bridge Test algorithm. The left two figures show Gaussian density with different σ and the right two figures show uniform density with different sizes (white = high density, black = low density).

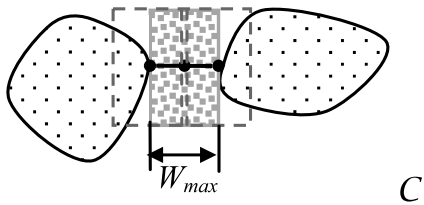

Figure 3. Maximum value of l, where the maximum permitted width of a narrow passage is equal to the width of the rigid body.

It is observed from Fig. 2 that a large value of l should be precluded as more computation time is required to identify a small volume of the permitted sampling region. In an extreme case, when $W_{\max}$ is equal to the maximum width of the rigid segment of the robot, there is no need to find a bridge in a region smaller than the volume of the robot (Fig. 3). Although a small value of l corresponds to a large volume of the permitted sampling region, time will be wasted in finding a long bridge across a wide area. In the other extreme case, when l is set to 2, the permitted sampling region of Bridge Test is equal to the whole planning region, resulting in a very long bridge across this extremely wide area, which would be of no benefit for the planning procedure. Generally speaking, l should not be too small or too large so that a narrow passage can be well defined. Clearly the best setting of l is problem-dependent, which makes it difficult to be chosen in advance. We empirically set the range of l to [10, 30] based on a series of independent experiments. A promising approach is to adjust l for each specific problem adaptively through online learning, so that the best value of l can be identified automatically in an adaptive way.

3. Second Stage: Triple-RRTs Path Planner

3.1. RRT-Connect Path Planner

RRTs, introduced by LaValle in 1998 [11], has been recognized as a very useful tool for designing single-query path planners. Starting at a given initial configuration, the classic RRTs incrementally grows a tree to explore the unknown configuration

Algorithm 2
RRT-Connect($q_{\text{init}}, q_{\text{goal}}$)

Step 1: Initialize the first tree structure $\mathcal{T}_{\text{a}}$ rooted at q_{init}. Initialize the second tree structure $\mathcal{T}_{\text{b}}$ rooted at q_{goal}.

Step 2: Randomly sample a configuration q_{rand} in $\mathcal{C}$.

Step 3: $\mathcal{T}_{\text{a}}$ extends one step to q_{rand}, generating a new configuration q_{new}. If q_{new} is collision-free, it is added as a child node to $\mathcal{T}_{\text{a}}$.

Step 4: $\mathcal{T}_{\text{b}}$ extends to q_{new} continuously until it either collides with obstacles or reaches q_{new}. Adding the new expanded configuration as a child node to $\mathcal{T}_{\text{b}}$.

Step 5: If $\mathcal{T}_{\text{b}}$ reaches q_{new}, $\mathcal{T}_{\text{a}}$ and $\mathcal{T}_{\text{b}}$ are connected, the algorithm terminates after a solution path is returned.

Step 6: Swap $\mathcal{T}_{\text{a}}$ and $\mathcal{T}_{\text{b}}$, Repeat Steps 2–5 until the maximum permitted number of iterations has been reached.

space. The probability that a point is selected for extension is proportional to the area of its Voronoi region, so that RRTs tends to rapidly orient to the unexplored regions of $\mathcal{F}$.

Classic RRTs only keeps a single tree from the initial configuration. The algorithm often attempts to extend towards an unexplored region instead of orienting to the goal configuration. Therefore, the algorithm may spend lots of computation time in sampling large open areas of the configuration space before finding a solution path. Two years later, an important improvement called RRT-Connect [13] was proposed to grow two trees from initial and goal configurations, respectively. Both a bidirectional expansion and a goal-oriented heuristic strategy are adopted to make two trees grow rapidly toward each other, resulting in a substantial improvement of performance.

The RRT-Connect algorithm is shown in Algorithm 2. Two trees, denoted by $\mathcal{T}_{\text{a}}$ and $\mathcal{T}_{\text{b}}$, are maintained at all times until they become connected and a solution path is returned. In each iteration, a configuration q_{rand} is randomly sampled to which the first tree extends for one step, generating a new node q_{new}, and then the second tree attempts to extend toward q_{new} as far as possible. Balanced extension is achieved by swapping two trees. As a result, the RRT-Connect algorithm can keep a fine balance between exploration and connection.

3.2. Triple-RRTs Path Planner

Although the RRT-Connect planner performs well in many applications, it still suffers from the drawback of uniform sampling strategy when there are narrow passages away from both the initial and goal configurations, which is illustrated by the so-called bug-trap example [1] in Fig. 4. The task is to move a robot with 2 translational d.o.f. from the left bug-trap obstacle to the right side. For both trees, they tend to explore wide open areas formed by the curved obstacles before finding the tiny opening, leaving the problem intractable (left-hand panel in Fig. 4).

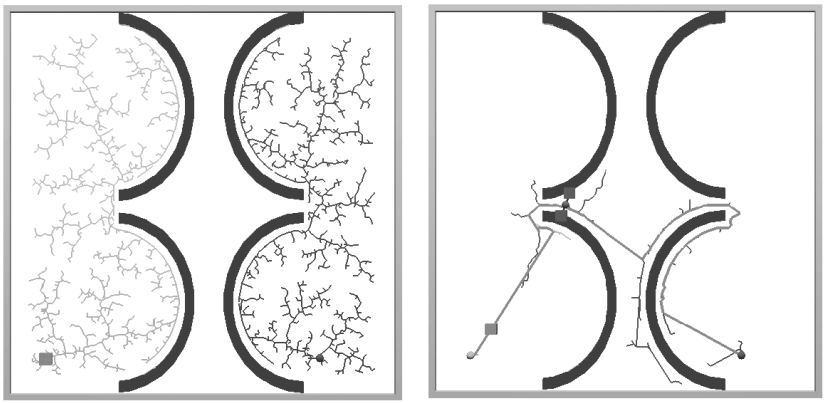

Figure 4. Bug-trap problem, where the RRT-Connect planner suffers from narrow passages.

Clearly, a prior understanding about the environment information will increase the possibility of passing through the tiny opening. Based on such consideration, we present two varieties of RRTs called Triple-RRTs in which the intermediate tree is grown from the global roadmap found by the first stage of the Bridge Test algorithm mentioned above. Both of the bridge endpoints are marked by the square block in Fig. 4. It is observed that fewer nodes have been extended before a valid solution path is found in the Triple-RRTs planner as the position of the opening is successful identified by our improved Bridge Test algorithm. The detailed description of the Triple-RRTs planners is discussed below.

Recall that we have obtained a sampled roadmap q_{m} through the improved Bridge Test algorithm described in Section 2. In order to apply global heuristics to RRTs exploration, the third intermediate tree $\mathcal{T}_{\mathrm{c}}$ is grown from the global landmark q_{m} except the previous two trees $\mathcal{T}_{\mathrm{a}}$ and $\mathcal{T}_{\mathrm{b}}$. That is why we call it Triple-RRTs planner. Since the root of $\mathcal{T}_{\mathrm{c}}$ lies in a narrow passage, it is expected to sample more configurations near the critical region so as to guide the tree through the identified narrow passage.

In the triple-trees planning framework, it is natural to ask which pair of trees should be selected to explore and connect. The simplest strategy is to connect q_{init} and q_{m} based on RRT-Connect algorithm to find the half path $Path_1$, while connect q_{m} and q_{goal} to find the other half path $Path_2$. The full path $Path$ is synthesized by joining $Path_1$ and $Path_2$ together. The strategy is called Simple-Triple-RRTs and shown in detail in Algorithm 3.

Note that the Simple-Triple-RRTs planner combines the RRT-Connect algorithm as a bidirectional expansion strategy to connect a pair of candidate trees. If BRIDGE_TEST() cannot find such a mid-point q_{m} to identify the narrow passage, the algorithm will be degraded to the standard RRT-Connect form. As a result, the Simple-Triple-RRTs planner generally performs better than the RRT-Connect algorithm.

Algorithm 3
Simple-Triple-RRTs($q_{\text{init}}, q_{\text{goal}}$)

Step 1: Call BRIDGE_TEST() to find the global roadmap q_{m}.

Step 2: Call RRT-Connect($q_{\text{init}}, q_{\text{m}}$) to generate one half path $Path_1$.

Step 3: Call RRT-Connect($q_{\text{m}}, q_{\text{goal}}$) to generate the other half path $Path_2$.

Step 4: Joint $Path_1$ and $Path_2$ together to synthesize the full path $Path$.

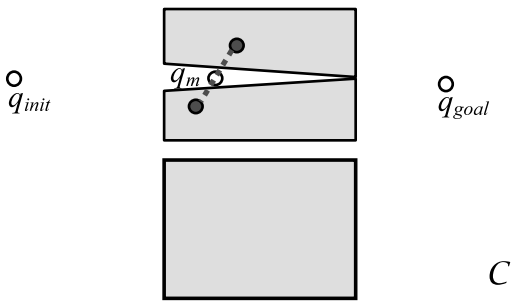

Figure 5. Narrow dead-end region.

However, the Simple-Triple-RRTs planner may be stuck in a so-called narrow dead-end passage with one end open and the other end closed, which is depicted in Fig. 5. As the Bridge Test algorithm is based on a test of local geometry of the configuration space, we cannot tell the difference between a narrow passage and a narrow dead-end. Although the drawback could be eliminated by using an orthogonal test in which an additional bridge is introduced to identify the opening of the narrow region, more computation time is required as it checks for collision 5 times (four endpoints plus one mid-point), while the original Bridge Test takes only 3 times in one loop. As a result, we attempt to eliminate the false-positive phenomena by improving the poor extending strategy of the Simple-Triple-RRTs planner.

Since the Simple-Triple-RRTs strategy requires the final path to go through the intermediate tree $\mathcal{T}_{\text{c}}$, the algorithm will return failure if any of the half paths cannot be generated successfully. Besides, if open wide corridors exist between q_{init} and q_{goal}, the performance of the algorithm will decrease as more time will be spent in exploring the identified difficult region, rather than choosing the path across the wide corridor which could be more preferable if it is connectable. The reasonable strategy is to equally choose a pair of candidate trees for exploration and connection at each time. Whenever a feasible path containing q_{init} and q_{goal} is found, the algorithm will terminate, regardless whether it passes through q_{m} or not. Therefore we design a balanced search algorithm called Balanced-Triple-RRTs, which is described in Algorithm 4.

Algorithm 4
Balanced-Triple-RRTs(q_{init}, q_{goal})

Step 1: Initialize the first tree structure $\mathcal{T}_{\text{a}}$ rooted at q_{init}. Initialize the second tree structure $\mathcal{T}_{\text{b}}$ rooted at q_{goal}. Clear the list $Path_1$, $Path_2$ and $Path$.

Step 2: Call BRIDGE_TEST() to find the global roadmap q_{m}. Initialize the third tree structure $\mathcal{T}_{\text{c}}$ rooted at q_{m}.

Step 3: Call Connect($\mathcal{T}_{\text{a}}$, $\mathcal{T}_{\text{b}}$) to extend one step from $\mathcal{T}_{\text{a}}$ to $\mathcal{T}_{\text{b}}$. If $\mathcal{T}_{\text{a}}$ and $\mathcal{T}_{\text{b}}$ are connected, the algorithm terminates after returning a solution path.

Step 4: If $Path_1$ does not exist, Call Connect($\mathcal{T}_{\text{a}}$, $\mathcal{T}_{\text{c}}$) to extend one step from $\mathcal{T}_{\text{a}}$ to $\mathcal{T}_{\text{c}}$. If $\mathcal{T}_{\text{a}}$ and $\mathcal{T}_{\text{c}}$ are connected, record the half path $Path_1$.

Step 5: If $Path_2$ does not exist, Call Connect($\mathcal{T}_{\text{c}}$, $\mathcal{T}_{\text{b}}$) to extend one step from $\mathcal{T}_{\text{c}}$ to $\mathcal{T}_{\text{b}}$. If $\mathcal{T}_{\text{c}}$ and $\mathcal{T}_{\text{b}}$ are connected, record the half path $Path_2$.

Step 6: If both $Path_1$ and $Path_2$ exist, joint $Path_1$ and $Path_2$ together to synthesize the full path $Path$. The algorithm terminates after returning a solution path $Path$.

Step 7: Swap $\mathcal{T}_{\text{a}}$ and $\mathcal{T}_{\text{b}}$, Repeat Steps 3–6 until the maximum permitted number of iterations has been reached.

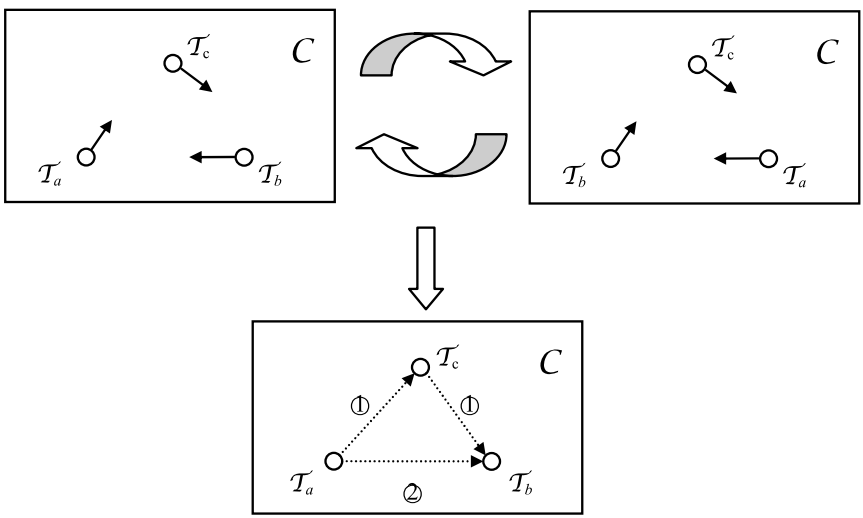

Figure 6. The illustration of the procedure of Balanced-Triple-RRTs extension.

Balanced-Triple-RRTs keeps three trees $\mathcal{T}_{\text{a}}$, $\mathcal{T}_{\text{b}}$, and $\mathcal{T}_{\text{c}}$ rooted at q_{init}, q_{goal} and the mid-point of the bridge q_{m}, respectively. Each tree has equal probability of extension by the alternate strategy. The algorithm is expected to return any valid path as long as the trees containing q_{init} and q_{goal} intersect at a particular point. The strategy is especially suitable for the situation that both narrow and wide open corridors coexist. The illustration of the Balanced-Triple-RRTs algorithm is shown in Fig. 6. It is observed that a fine balance of tree expansion is achieved to return either

'path 1', when the intermediate tree is easy to pass, or 'path 2', when it is difficult to connect the intermediate tree to both the initial and goal tree. In conclusion, the algorithm takes advantage of the Bridge Test algorithm as much as possible, while eliminating the possible drawbacks of the Bridge Test algorithm during the procedure of RRTs extensions.

4. Experimental Results

We implemented both the Simple- and Balanced-Triple-RRTs planners based on the Motion Strategy Library (http://msl.cs.uiuc.edu/msl/) developed by Professor Steven M. LaValle from the University of Illinois at Urbana Champaign. We have compared the performances of the three RRTs algorithms described in Algorithms 2–4. For each of the experiments, we show the running times (Times), the total number of nodes in the trees (Nodes) and the total number of collision detection calls (CD Calls) averaged over 30 runs. As an important preprocessing stage, the performance of the Bridge Test algorithm is listed separately. According to the analysis mentioned in Section 2.3, the parameter l in to Bridge Test is set to be 20 *a priori*. All of the experiments are performed on a 3.0 GHz Pentium PC with 1.0 GB memory.

The first experiment is for a rigid segment robot with 2 translational d.o.f. and 1 rotational d.o.f. The robot has to move from the large open area on the left side, through a narrow zigzag corridor, to the right side. The tree structure, as well as the solution path, is shown in Fig. 7. The performance comparison of the three RRTs algorithms is shown in Table 1. As expected, the Triple-RRTs planners can effectively identify and pass through the narrow passage in about 2–3 s. They give about 5 times improvement in the running time over the RRT-Connect planner in this experiment. Note that the illustrations of the structures of expanded trees are projected from 3-D configuration space to the 2-D robot work space.

The second experiment, shown in Fig. 8, is the bug-trap environment for the same robot. The robot has to get through two tiny openings to reach the goal position. It is observed from Table 1 that the Triple-RRTs planners improve running time

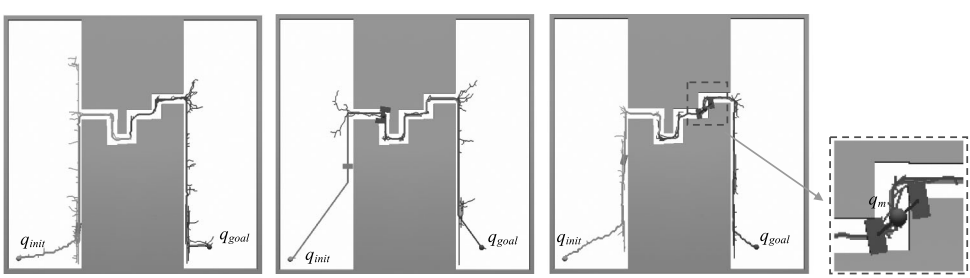

Figure 7. A 2-D example. A 3-d.o.f. robot has to move through the zigzag corridor from the left side to the right side. The first three figures show tree structures generated by RRT-Connect, Simple-Triple-RRTs and Balanced-Triple-RRTs algorithms, respectively.

Table 1.
Performance comparisons of the RRT-Connect and Triple-RRTs algorithms

Experiment	Planners		Time (s)	Nodes	CD calls
1	RRT-Connect		14.36	782	14602
	Simple-Triple-RRTs	Bridge test	0.039	—	318
		Planning	2.53	393	6248
	Balanced-Triple-RRTs	Bridge test	0.015	—	264
		Planning	3.22	469	8268
2	RRT-Connect		105.56	3993	127290
	Simple-Triple-RRTs	Bridge test	0.031	—	471
		Planning	0.80	220	3837
	Balanced-Triple-RRTs	Bridge test	0.016	—	384
		Planning	0.60	180	3209
3	RRT-Connect		116.95	4867	255717
	Simple-Triple-RRTs	Bridge test	0.016	—	51
		Planning	2.11	319	21624
	Balanced-Triple-RRTs	Bridge test	0.012	—	46
		Planning	1.67	190	12574
4	RRT-Connect		62.26	593	13321
	Simple-Triple-RRTs	Bridge test	18.69	—	5763
		Planning	29.84	398	5622
	Balanced-Triple-RRTs	Bridge test	20.38	—	6146
		Planning	32.36	452	6088

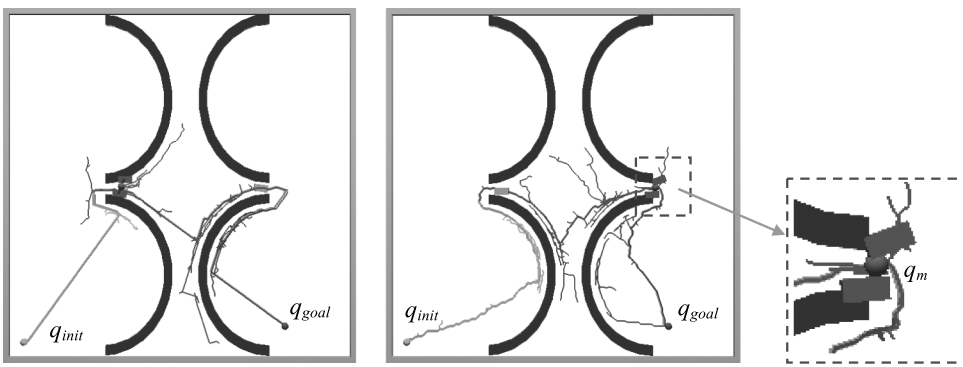

Figure 8. Bug-trap example. A 3-d.o.f. robot has to move from the left bug-trap side to the right side. The two figures on the left show tree structures generated by Simple-Triple-RRTs and Balanced-Triple-RRTs, respectively.

more than 120 times over the RRT-Connect planner. Interestingly, the performance of Balanced-Triple-RRTs planner is slightly better than that of the Simple-Triple-RRTs planner. The reason is partly because balanced explorations among three trees rather than two trees may be of benefit to find underlying connectivity in high-

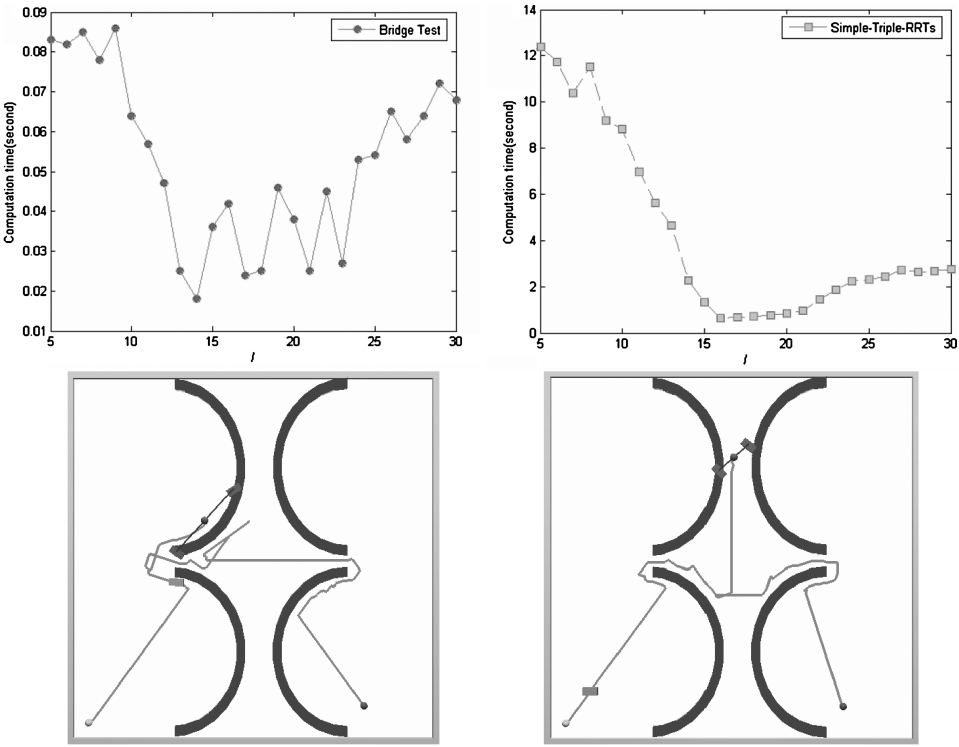

Figure 9. Performance comparisons of both Bridge Test and the Simple-Triple-RRTs planner for different value of l. The bottom two figures show two types of the bridge points in worst cases when l is set to 5 and 10, respectively.

dimensional configuration space. The illustrations of the structures of expanded trees are also projected from 3-D configuration space to the 2-D robot work space.

In the bug-trap experiment, the size of the environment is 100×100, and the size of the robot is 4×2. According to equation (3), the valid range of l is calculated to be [2, 100]. We select different values of l from 5 to 30, and test the performances of both Bridge Test and the Simple-Triple-RRTs planner. The result is shown in Fig. 9. When the value of l is small, an invalid collision-free endpoint will be sampled with a high probability in a wide permitted region of uniform random sampling, so that the performance of Bridge Test is low. The performance of Simple-Triple-RRTs is greatly affected by the invalid roadmap found in some difficult regions (bottom two panels of Fig. 9), which will not help increase the connectivity of the global $\mathcal{C}$ space. The performance is also affected slightly by a large value of l. For the Bridge Test algorithm, it requires more computation time to identify either a short bridge or a long bridge in $\mathcal{C}$ space. For the Simple-Triple-RRTs planner, the performance does not decrease so much once a valid global roadmap that locates in the opening region has been returned successfully. It is observed that the appropriate range for l in this experiment is [15, 22], in accordance with our expectation.

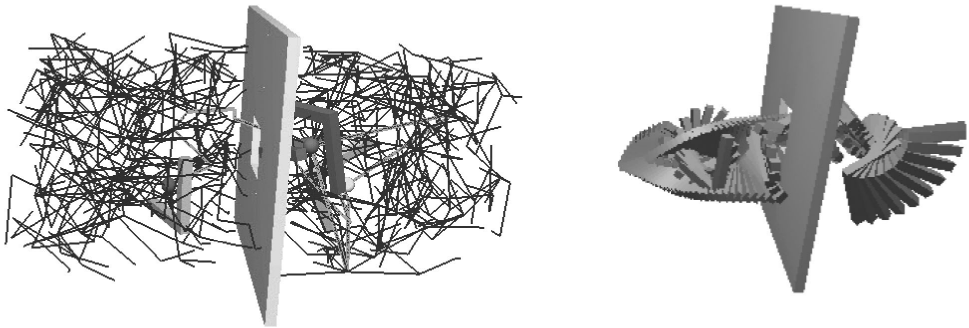

Figure 10. A 3-D example. An L-shaped 6-d.o.f. robot has to move through a wall with a small hole in the center. The left figure shows the robot initial and goal positions, Bridge Test points, as well as the tree structure generated by the Simple-Triple-RRTs planner. The right figure shows one solution path found by the Balanced-Triple-RRTs planner.

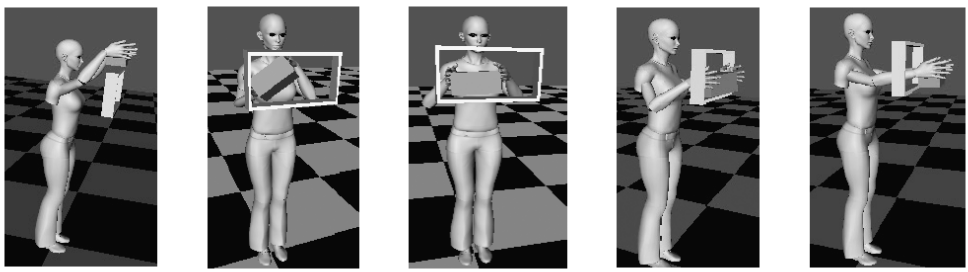

Figure 11. Manipulation planning example. The virtual human has to hold a box to move through a manipulation window. The figure shows the animation snapshots in 1, 87, 132, 158 and 235 frames, respectively.

The third experiment, shown in Fig. 10, is for an L-shaped 6-d.o.f. robot navigating in the 3-D environment. Here, the robot, consisting of two perpendicular blocks, has to pass through a broad wall with a small hole in the center. Both of the blocks have the dimension $30 \times 4 \times 4$, while the hole has the dimension $20 \times 20 \times 4$. The robot's movement is very restricted when it is inside the small hole. Again we compare the three RRTs algorithms and the results are shown in Table 1. There is an improvement of two orders of magnitude in the running time for this experiment, the same as the bug-trap experiment. Note that the performance of the Balanced-Triple-RRTs planner is still slightly better than that of the Simple-Triple-RRTs planner, implying that the Balanced-Triple-RRTs planner may generate more samples in the narrow passage so as to pass through the narrow passage more efficiently than the Simple-Triple-RRTs planner.

The final experiment is a manipulation planning problem that is shown in Fig. 11. A virtual human has to hold a box to move through a manipulation window. Each articulated arm of the virtual human contains 7 d.o.f., and the manipulated box has 6 d.o.f. The total dimension of the configuration space in this problem adds up to 20, much higher than all of the above experiments. Furthermore, there is a closed-chain

constraint between both arms and the manipulated object. The inverse kinematics toolkit IKAN [29] is applied to account for the closed kinematic constraints. The joint angles of both arms are calculated based on the spatial posture of the manipulated object. The result is shown in Table 1. It is observed that the performance improvement is not so great as in the above experiments, because the configuration space is highly constrained by kinematics and the joint rotation limits. It is observed from Table 1 that much computation time is spent in Bridge Test in such a high-dimensional configuration space. The intention of showing this example is to demonstrate that the Triple-RRTs planner can also be applied to solve multi-d.o.f. robot motion planning problems with both open and closed chains, which is of great importance to the robotics and computer graphics community.

5. Conclusions

We have presented two varieties of Triple-RRTs planners based on the improved Bridge Test algorithm to address the multi-d.o.f. robot path planning problem in narrow passages. The proposed Triple-RRTs planners take advantage of the global heuristics of PRM and local connection of the RRTs algorithm. Our algorithm proceeds in two stages. The first stage improves the Bridge Test algorithm to find a global landmark that identifies the critical region efficiently. The second stage grows triple RRTs trees from the queried configurations and the global landmark. The bidirectional RRT-Connect algorithm is applied to search for a local connection among these trees, guiding the path through a narrow passage rapidly. Compared to previous single-query path planning methods, the two-stage strategy is more effective in finding a global feasible path as a fine balance is kept between global guidance and local connection. The approach is simple, general and experiments demonstrate that it achieves high performance for a wide class of multi-d.o.f. robot path planning problems.

Even though the proposed method has numerous advantages in environments with narrow passages, several unresolved issues remain for further study.

A major problem is the optimum value of the parameter l. Although we give an approximate range for l through a large number of independent experiments, the optimal value is certainly problem-specific, leaving it intractable to be determined *a priori*. The reasonable value is expected to be chosen based on on-line reinforcement learning that could incrementally understand the local geometry of narrow passages.

In addition, the method can be generalized to multi-RRTs planner. Since the RRT-Connect algorithm is considered as an effective local path planner given a pair of queried configurations, a global path planning algorithm depending on the multi-RRTs framework can be built based on a variety of sampled roadmaps in critical regions, which would clearly improve the generality of the planner in more complex environments. However, there are two main open problems:

(i) How many sampled configurations are enough to identify all of the narrow passages? How to efficiently evaluate the validity of sampled configuration found by the Bridge Test? The more sampled configurations, the greater the understanding about the configuration space. However, poor sampled roadmaps will reduce the performance of the planner dramatically. We intend to apply the clustering technique to generate a clustering center of the sampled roadmaps to identify the critical regions. As the Bridge Test algorithm checks collisions 3 times in one loop, it is not reasonable to call the Bridge Test frequently. The optimal number of sampled configurations should be calculated for general path planning problems.

(ii) How to efficiently merge a pair of trees? If two trees are difficult to merge, does it imply that one tree lies in the disconnected region so as to be deleted from the multi-RRTs framework, or on the contrary, both of the trees locate in narrow passages that should be connected frequently in the successive iterations? Thus, a well-posed merging strategy should be designed for general single-queried path planning problems.

Both of the above issues remain areas for further study.

Acknowledgements

We would like to thank Steven M. LaValle for providing the source code of the Motion Planning Library. This work was supported in part by the National High Technology Research and Development Program of China (863 Program) under grant 2006AA110114, and National Natural Science Foundation of China (NSFC) under grants 90820302 and 60774076. The authors are very grateful to anonymous reviewers for valuable suggestions that have helped strengthen the paper.

References

1. S. M. LaValle, *Planning Algorithms*. Cambridge University Press, Cambridge (2006).
2. J.-C. Latombe, *Robot Motion Planning*. Kluwer, Norwell, MA (1991).
3. L. E. Kavraki, P. Svestka, J.-C. Latombe and M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. Robotics Automat*. **12**, 566–580 (1996).
4. J. Cortés and T. Siméon, Sampling-based motion planning under kinematic loop-closure constraints, in: *Proc. 6th Int. Workshop on Algorithmic Foundations of Robotics*, Zeist, pp. 59–74 (2004).
5. L. Han and N. M. Amato, A kinematics-based probabilistic roadmap method for closed chain systems, in: *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch and D. Rus (Eds), pp. 233–246. A. K. Peters, Wellesley, MA (2001).
6. J. Yakey, S. M. LaValle and L. E. Kavraki, Randomized path planning for linkages with closed kinematic chains, *IEEE Trans. Robotics Automat*. **17**, 951–958 (2001).
7. P. Svestka and M. H. Overmars, Coordinated motion planning for multiple car-like robots using probabilistic roadmaps, *IEEE Trans. Robotics Automat*. **11**, 1631–1636 (1995).

8. S. Sekhavat, P. Svestka, J.-P. Laumond and M. H. Overmars, Multilevel path planning for nonholonomic robots using semiholonomic subsystems, *Int. J. Robotics Res.* **17**, 840–857 (1998).
9. E. Mazer, J. M. Ahuactzin and P. Bessière, The Ariadne's Clew algorithm, *J. Artif. Intell.* **9**, 295–316 (1998).
10. D. Hsu, J.-C. Latombe and R. Motwani, Path planning in expansive configuration spaces, *Int. J. Comput. Geometry Applic.* **4**, 495–512 (1999).
11. S. M. LaValle, Rapidly-exploring random trees: a new tool for path planning, *TR 98-11*, Computer Science Department, Iowa State University (1998).
12. S. M. LaValle and J. J. Kuffner, Rapidly-exploring Random Trees: progress and prospects, in: *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch and D. Rus (Eds), pp. 293–308. A. K. Peters, Wellesley, MA (2001).
13. J. J. Kuffner and S. M. LaValle, RRT-connect: an efficient approach to single-query path planning, in: *Proc. Int. Conf. on Robotics and Automation*, San Francisco, CA, pp. 995–1001 (2000).
14. S. Kagami, J. Kuffner, K. Nishiwaki, K. Okada, M. Inaba and H. Inoue, Humanoid arm motion planning using stereo vision and RRT search, in: *Proc. Int. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, pp. 2167–2172 (2003).
15. M. Kallmann, A. Aubel, T. Abaci and D. Thalmann, Planning collision-free reaching motions for interactive object manipulation and grasping, in: *Proc. Eurographics*, Granada, pp. 313–322 (2003).
16. A. Yershova and S. M. LaValle, Planning for closed chains without inverse kinematics. Available online at http://msl.cs.uiuc.edu/~yershova/icra2007/paper.pdf.
17. E. Frazzoli, M. A. Dahleh and E. Feron, Real-time motion planning for agile autonomous vehicles, *AIAA J. Guid. Control* **25**, 116–129 (2002).
18. F. Lamiraux, E. Ferre and E. Vallee, Kinodynamic motion planning: connecting exploration trees using trajectory optimization methods, in: *Proc. Int. Conf. on Robotics and Automation*, New Orleans, LA, pp. 3987–3992 (2004).
19. A. Yershova, L. Jaillet, T. Simon and S. M. LaValle, Dynamic domain RRTs: efficient exploration by controlling the sampling domain, in: *Proc. Int. Conf. on Robotics and Automation*, Barcelona, pp. 3856–3861 (2005).
20. S. R. Lindemann and S. M. LaValle, Incrementally reducing dispersion by increasing Voronoi Bias in RRTs, in: *Proc. Int. Conf. on Robotics and Automation*, New Orleans, LA, pp. 3251–3257 (2004).
21. T. Y. Li and Y. C. Shie, An incremental learning approach to motion planning with roadmap management, in: *Proc. Int. Conf. on Robotics and Automation*, Washington, DC, pp. 3411–3416 (2002).
22. M. Strandberg, Augmenting RRT-planners with local trees, in: *Proc. Int. Conf. on Robotics and Automation*, New Orleans, LA, pp. 3258–3262 (2004).
23. D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwari and S. Sorkin, On finding narrow passages with probabilistic roadmap planners, in: *Proc. 3rd Int. Workshop on Algorithmic Foundations of Robotics*, Houston, TX, pp. 141–153 (1998).
24. N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones and D. Vallejo, OBPRM: an obstacle-based PRM for 3D workspaces, in: *Proc. 3rd International Workshop on Algorithmic Foundations of Robotics*, Houston, TX, pp. 155–168 (1998).
25. N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones and D. Vallejo, Choosing good distance metrics and local planners for probabilistic roadmap methods, *IEEE Trans. Robotics Automat.* **16**, 442–447 (2000).

26. S. A. Wilmarth, N. M. Amato and P. F. Stiller, MAPRM: a probabilistic roadmap planner with sampling on the medial axis of the free space, in: *Proc. Int. Conf. on Robotics and Automation*, Detroit, MI, pp. 1024–1031 (1999).
27. V. Boor, M. H. Overmars and A. F. van der Stappen, Gaussian sampling for probabilistic roadmap planners, in: *Proc. Int. Conf. on Robotics and Automation*, Detroit, MI, pp. 1018–1023 (1999).
28. Z. Sun, D. Hsu, T. Jiang, H. Kurniawati and J. H. Reif, Narrow passage sampling for probabilistic roadmap planning, *IEEE Trans. Robotics* **21**, 1105–1115 (2005).
29. D. Tolani, A. Goswami and N. I. Badler, Real time inverse kinematics techniques for anthropomorphic limbs, *Graph. Models* **62**, 353–388 (2000).

About the Authors

Wei Wang received the BS degree in Control Engineering from the College of Mechatronics and Automation, National University of Defense Technology, Changsha, P. R. China, in 2003. He is now a PhD candidate in Control Engineering at the National University of Defense Technology. His current research interests are robotics, virtual humans, motion planning and algorithms.

Xin Xu received the BS and PhD degrees in Control Engineering from the National University of Defense Technology, Changsha, P. R. China, in 1996 and 2001, respectively. From 2003 to 2004, he was a Postdoctoral Fellow at the School of Computer, National University of Defense Technology. He has been a Visiting Scholar at the Hong Kong Polytechnic University, Hong Kong, P. R. China and the University of Strathclyde, UK, in 2006 and 2007, respectively. Currently, he is an Associate Professor at the College of Mechatronics and Automation, National University of Defense Technology. He received the excellent PhD dissertation award from Hunan Province, P. R. China, in 2004, and the Fork Ying Tong Youth Teacher Fund of China, in 2008. He is a Member of the IEEE Computational Intelligence Society and the IEEE Control Systems Society. His research interests include reinforcement learning, data mining, learning control, robotics, autonomic computing and computer security.

Yan Li received the BS, MS and PhD degrees in Control Engineering from the National University of Defense Technology, Changsha, P. R. China, in 1995, 1998 and 2002, respectively. He has been an Associate Professor at the College of Mechatronics and Automation, National University of Defense Technology, since 2005. His research interests include computer graphics, robotics, virtual reality and teleoperation.

Jinze Song received the BS and MS degrees in Control Engineering from the National University of Defense Technology, Changsha, P. R. China, in 2001 and 2004, respectively. He is now a PhD candidate in Control Engineering at the National University of Defense Technology. His current research interests are predictive control and robotics.

Hangen He is the Professor of Intelligent Control at the National University of Defense Technology. He received the BS degree from the Department of Physics, Harbin Institute of Technology, Harbin, P. R. China, in 1969. His research interests include robotics, autonomous land vehicles (ALVs), artificial intelligent and computer graphics. He is the Advisory Chairman of the Chinese Association of Intelligent Robotics. He is one of the leading scientists in China in the field of ALVs. Cooperating with the First Automobile Workshop of China, he and his research team designed and manufactured the first ALV car, HongQi, in China, in 2004. The stable speed of HongQi on the highway can reach 130 km/h without human interference and the peak speed can reach as high as 170 km/h.