Expert Systems With Applications 152 (2020) 113425

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

PQ-RRT*: An improved path planning algorithm for mobile robots

Yanjie Li[a], Wu Wei[a,*], Yong Gao[a], Dongliang Wang[a], Zhun Fan[b,c,d,e]

[a] School of Automation Science and Engineering, South China University of Technology, Guangzhou 510641, China
[b] Department of Electronic and Information Engineering, Shantou University, Shantou 515063, China
[c] Key Lab of Digital Signal and Image Processing of Guangdong Province, Shantou University, Shantou 515063, China
[d] Key Laboratory of Intelligent Manufacturing Technology, Shantou University, Ministry of Education 515063, China
[e] State Key Lab of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 43003, China

ARTICLE INFO

Article history:
Received 21 September 2019
Revised 25 January 2020
Accepted 30 March 2020
Available online 6 April 2020

Keywords:
Path planning
Sampling-based algorithms
Rapidly-exploring random tree (RRT)
Optimal path planning

ABSTRACT

During the last decade, sampling-based algorithms for path planning have gained considerable attention. The RRT*, a variant of RRT (rapidly-exploring random trees), is of particular concern to researchers due to its asymptotic optimality. However, the limits of the slow convergence rate of RRT* makes it inefficient for applications. For the purposes of overcoming these limitations, this paper proposes a novel algorithm, PQ-RRT*, which combines the strengths of P-RRT* (potential functions based RRT*) and Quick-RRT*. PQ-RRT* guarantees a fast convergence to an optimal solution and generates a better initial solution. The asymptotic optimality and fast convergence of the proposed algorithm are proved in this paper. Comparisons of PQ-RRT* with P-RRT* and Quick-RRT* in four benchmarks verify the effectiveness of the proposed algorithm.

1. Introduction

Mobile robots are gaining more and more attention as intelligent systems. Much attention has been devoted to mobile robots since they can increase productivity and provide various conveniences. Mobile robots with autonomous navigation capabilities are critical to machine intelligence. The basis of navigation is path planning (Huang, Li, Jiang, & Cheng, 2019; Lee, Shin, & Chae, 2018; Majeed & Lee, 2019), which consists of finding feasible paths from the start state to the goal state without colliding with any obstacles. Applications of path planning algorithms include but are not limited to industrial automation (Beyer, Jazdi, Göhner, & Yousefifar, 2015; Pandini, Spacek, Neto, & Junior, 2017), graphical animation (Liu & Badler, 2003), autonomous exploration (Atanacio-Jiménez et al., 2011), medical (Taylor, Menciassi, Fichtinger, Fiorini, & Dario, 2016; Valencia-Garcia, Martinez-Béjar, & Gasparetto, 2005) and robot navigation (González, Pérez, Milanés, & Nashashibi, 2015).

The research on path planning is in full swing. Currently, path planning algorithms mainly include geometric algorithms, artificial potential field methods, grid-based searches, and sampling-based algorithms. Typical representatives of geometric algorithms include visibility graph (Alexopoulos & Griffin, 1992; Asano, Asano, Guibas, Hershberger, & Imai, 1985; Maekawa, Noda, Tamura, Ozaki, & Machida, 2010) and cell decomposition methods (Brooks & Lozano-Perez, 1985). However, these algorithms are mostly limited to low-dimensional path planning problems. Artificial potential fields (APF) (Khatib, 1986), a method first proposed by Khatib, should be given special consideration. APF assumes that there is a virtual force consisting of the repulsive force of the obstacles and the attractive force of the goal region. The system proceeds according to their joint force. Although the operation is easy, APF suffers from the problem of local minima (Koren & Borenstein, 1991). When applying graph theory to a discretized state space, grid-based searches assume that each state corresponds to a grid point. A well-known algorithm in grid-based algorithms is A* (Hart, Nilsson, & Raphael, 1968; Koenig, Likhachev, & Furcy, 2004; Stentz, 1997). Although it can guarantee resolution completeness and resolution optimality, computation time and memory space will grow exponentially in dimensions.

Sampling-based algorithms have gained a considerable amount of attention as the result of their superior performance in high-dimensional state spaces. These algorithms provide probabilistic completeness. Probabilistic completeness means that if feasible path exists, the probability that the algorithm cannot find the path

* Corresponding author.
E-mail addresses: 1073889317@qq.com (Y. Li), weiwu@scut.edu.cn (W. Wei), andygao_scut@163.com (Y. Gao), 757613634@qq.com (D. Wang), zfan@stu.edu.cn (Z. Fan).

https://doi.org/10.1016/j.eswa.2020.113425
0957-4174/

tends towards zero as the number of samples approaches infinity. Undisputed, the most widely used and influential algorithms to date are probabilistic roadmaps (PRMs) (Amato & Wu, 1996; Kavraki & Latombe, 1994; Kavraki, Svestka, Latombe, & Overmars, 1996) and rapidly-exploring Random Trees (RRTs) (LaValle, 1998). The PRM algorithm is a multiple-query method, which is suitable for problems with prior knowledge of the environment. In practice, an environment is not necessarily known in advance, and the multiple-queries methods do not always perform well. As single-query counterparts to PRMs, RRTs have won the attention of the research community, due to their greater efficiency in practical applications. Due to increased attention, a large number of RRT-variants have emerged in the last decade. Introducing a greedy heuristic to RRT, a double tree algorithm, RRT-connect, was proposed. Unlike RRT, RRT-connect (Kuffner & LaValle, 2000) maintains two trees at the same time: one from the initial state and the other from the goal state. However, neither RRT and RRT-connect consider the cost of the found solution, so both cannot ensure optimality. Inspired by the above idea, Karaman et al. (Karaman & Frazzoli, 2011) proposed the RRT* algorithm as an optimal variant of RRT. Different from RRT, RRT* adopts optimisation modules, the ChooseParent and Rewire procedures, ensuring the asymptotic optimality. An algorithm is said to be asymptotically optimal if the probability of finding the optimal solution approaches one when the number of samples approaches infinity. RRT* is a milestone in the development of RRT.

Although finding the optimal path solution is already a challenging task, ensuring fast convergence is also important for path planning. For the purpose of overcoming the slow convergence rate of RRT*, a lot of work has gradually unfolded. Enlightened by the idea of visibility graph technology, RRT*-Smart (Islam, Nasir, Malik, Ayaz, & Hasan, 2012) introduces an intelligent sampling method to RRT*. RRT*-Smart accelerates the convergence rate and obtains an optimum or a near optimum solution. However, the quality of the solution obtained by RRT*-Smart greatly depends on the initial solution, which reduces the probability of finding a different homotopy class, thus violating the assumption of uniform sampling of RRT*. As we know, as the solution optimisation process proceeds, the number of nodes will increase infinitely. Therefore, the implementation of RRT* will be difficult in a computing system with limited memory. In order to use the memory more efficiently, Olzhas et al. put forward RRT* FN (Adiyatov & Varol, 2013), an algorithm with a fixed number of nodes. When the number of nodes exceeds a given value, it is important to remove nodes that are not useful in reducing the cost. In addition to the above methods, it is worth noting the sampling heuristic known as node rejection (Akgun & Stilman, 2011; Ferguson & Stentz, 2006). Informed RRT* (Gammell, Srinivasa, & Barfoot, 2014), inspired by node rejection, uses a direct sampling method that samples in a hyper-ellipsoid. However, the algorithm will no longer be applicable when the associated prolate hyperspheroid is larger than the domain of the planning problem.

A fundamental reason for the slow convergence rate of RRT* is its pure exploration, while APF (Khatib, 1986) suffers from the problem of local minima due to its pure exploitation. Based on the above ideas, Qureshi et al. proposed an improved algorithm, P-RRT* (Qureshi & Ayaz, 2016), which incorporates the APF into RRT*. It is the addition of APF that provides a direction for exploration, giving P-RRT* a faster convergence than RRT*. P-RRT* introduces APF to achieve a trade-off between exploration and exploitation, which is worth noting. Accelerating the convergence rate of RRT* can not only start from the guided sampling process, but it can also be optimised from the structure of RRT* itself. In this regard, Jeong et al. first point out that use of triangular inequality to improve the ChooseParent and Rewire procedures, proposing Quick-RRT* (Jeong, Lee, & Kim, 2019). Compared with RRT*, Quick-RRT* allows a faster rate of convergence. Since Quick-RRT* is a tree-extending algorithm, any sampling strategy or graph-pruning algorithm can be combined with Quick-RRT*.

This paper proposes a novel algorithm, PQ-RRT*, for the optimal path planning mobile robots. Compared with P-RRT* and Quick-RRT*, PQ-RRT* generates a better initial solution and a fast convergence to optimal solution. A theoretical proof is given for the completeness, asymptotic optimality and faster convergence of the proposed algorithm. In addition, PQ-RRT* has the same computational complexity as P-RRT* and Quick-RRT*. Comparative simulations are performed according to four benchmarks, which validate the effectiveness of the proposed algorithm.

The remaining sections of this paper are outlined as follows. Section 2 addresses the problem definition. Section 3 introduces the relevant prerequisites for the proposed algorithm. Section 4 explains the proposed algorithm. Section 5 presents analysis of completeness, optimality and computational complexity. Section 6 provides the simulation results. Section 7 summarizes the main contributions and discusses some future research directions.

2. Problem definition

This section presents three motion planning problems to be solved. Let $X \subseteq \mathbb{R}^d$ be the configuration space, where $d \in \mathbb{N}$, $d \geq 2$. Let $X_{obs} \subset X$ be the obstacle region, and denote the obstacle-free space as $X_{free} = cl(X \backslash X_{obs})$, where $cl(\cdot)$ denotes the closure of a set. x_{init} and X_{goal} are the initial configuration and the goal region, respectively. A continuous function σ: [0, 1]$\mapsto X$ is called a path, if it has bounded variation. The path is collision-free, if $\sigma(\tau) \in X_{free}$ for all $\tau \in$ [0, 1].

A path planning problem is to find a collision-free path σ: [0, 1]$\mapsto X_{free}$ that starts from the initial configuration $\sigma(0) = x_{init}$ and reaches the goal region $\sigma(1) \in X_{goal}$ and $\sigma(\tau) \in X_{free}$ for all $\tau \in$ [0, 1]. If a path σ: [0, 1]$\mapsto X$ is a collision-free path, $\sigma(0) = x_{init}$ and $\sigma(1) \in X_{goal}$, then it is called a feasible path. Given a triplet $\{x_{init}, X_{obs}, X_{goal}\}$, path planning problem is to find a feasible path. Problem 1 presents the feasibility problem of path planning.

Problem 1 (Feasible Path Planning)**.** Given a triplet $\{x_{init}, X_{obs}, X_{goal}\}$, find a feasible path, if one exists. Report failure if no such solution exists.

Let Σ denote the set of all paths, and $\Sigma_{feasible}$ is a set of all feasible paths. Let $c(\cdot)$ be the cost function in terms of Euclidean distance function. Problem 2 formalizes the optimality problem of path planning.

Problem 2 (Optimal Path Planning)**.** Given a triplet $\{x_{init}, X_{obs}, X_{goal}\}$ and a cost function $c : \Sigma \rightarrow \mathbb{R}_{\geq 0}$, find a feasible path σ^* such that $c(\sigma^*) = \min\{c(\sigma) : \sigma \in \Sigma_{feasible}\}$. Report failure if no such solution exists.

Let $t \in \mathbb{R}$ denote the time required by the algorithm to find a set of all feasible paths $\Sigma_{feasible}$. The fast path planning stated in Problem 3 demonstrates that optimal path solution must be found in least possible time.

Problem 3 (Fast Path Planning)**.** Find the optimal path solution in least possible time $t \in \mathbb{R}$.

3. Related work

This section first introduces RRT*, which forms the basis of P-RRT* and Quick-RRT* algorithms, and then it explains P-RRT* and Quick-RRT* subsequently. The above two algorithms are the cornerstones of the proposed PQ-RRT* algorithm.

3.1. RRT*

This section formally introduces RRT*, which is an incremental sampling-based motion planning algorithm. RRT* guarantees asymptotic optimality, that is, an almost-sure convergence to optimal solution. Before showing the RRT* algorithm, a brief description of RRT will be provided. The inputs of RRT consist of the initial state, the goal state (region) and the environment. The output is a graph including a feasible path. In each iteration, a sample x_{rand} is selected randomly from X_{free}, then the closest vertex $x_{nearest}$ to the sample x_{rand} is found in terms of distance metric. The graph verifies whether there is a feasible local path σ_{local} from $x_{nearest}$ to x_{rand}. If so, the sample x_{rand} and the local path σ_{local} are added to the tree. The above steps are repeated until a feasible path is found or the stop criteria are met. Next, we will briefly describe RRT*, which is shown in Algorithm 1.

Algorithm 1 RRT*.

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;
for $i = 1$ to n **do**
  $x_{rand} \leftarrow SampleFree(i)$;
  $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
  $\sigma \leftarrow Steer(x_{nearest}, x_{rand})$;
  if $CollisionFree(\sigma)$ **then**
    $X_{near} \leftarrow Near(V, x_{rand}, min\{\gamma_{RRT*}(\log(card(V)))/card(V)^{1/d}, \eta\})$;
    $(x_{parent}, \sigma_{parent}) \leftarrow ChooseParent(X_{near}, x_{nearest}, \sigma)$;
    $V \leftarrow V \cup \{x_{rand}\}$;
    $E \leftarrow E \cup (x_{parent}, x_{rand})$;
    $G \leftarrow Rewire(G, x_{rand}, X_{near})$;
  end if
end for
return $G = (V, E)$;

Unlike RRT, RRT* adopts two optimisation procedures, ChooseParent and Rewire. In the ChooseParent procedure, RRT* searches the X_{near}, (a set of vertices in a hypersphere of a specific radius centered at x_{rand}), to find the optimum. Through the optimisation, a path has the lowest cost from the root x_{init} to x_{rand}. The ChooseParent procedure is outlined in Algorithm 2. After adding x_{rand} to

Algorithm 2 ChooseParent(X_{near}, $x_{nearest}$, x_{rand}, $\sigma_{nearest}$).

$x_{\min} \leftarrow x_{nearest}$;
$\sigma_{\min} \leftarrow \sigma_{nearest}$;
$c_{\min} \leftarrow Cost(x_{nearest} + Cost(\sigma_{nearest}))$;
for each $x_{near} \in X_{near}$ **do**
  $\sigma \leftarrow Steer(x_{near}, x_{rand})$;
  $c \leftarrow Cost(x_{near} + Cost(\sigma))$;
  if $c < c_{\min}$ **then**
    if $CollisionFree(\sigma)$ **then**
      $x_{\min} \leftarrow x_{near}$;
      $\sigma_{\min} \leftarrow c$;
    end if
  end if
end for
return $(x_{\min}, \sigma_{\min})$;

the tree, RRT* tries to optimise the cost of the element of X_{near} through x_{rand}. If the local path σ_{local} from x_{rand} to any element of X_{near}, x_{near}, is collision-free, and the local path σ_{local} has a lower cost than the current path, then the parent of x_{near} is replaced with x_{rand}. Algorithm 3 presents the Rewire procedure. Following this, the primitive procedures involved in RRT* are briefly explained.

- SampleFree: It returns a random sample from X_{free}.

Algorithm 3 Rewire(G,x_{rand}, X_{near}).

for each $x_{near} \in X_{near}$ **do**
  $\sigma \leftarrow Steer(x_{rand}, x_{near})$;
  if $Cost(x_{rand}) + Cost(\sigma) < Cost(x_{near})$ **then**
    if $CollisionFree(\sigma)$ **then**
      $x_{parent} \leftarrow Parent(x_{near})$;
      $E \leftarrow (E \backslash \{(x_{parent}, x_{near})\}) \cup \{(x_{rand}, x_{near})\}$;
    end if
  end if
end for
return $G = (V, E)$;

- Nearest: It returns the closest vertex from V in the graph G = (V, E) in terms of a given distance metric. In this paper, we use Euclidean distance function.
- Near: Given a graph G = (V, E) and a configuration x, it returns the set of neighboring vertices of the sample x. The set is the vertices contained in a ball of a radius r centered at x.
- CollisionFree: It checks whether the local path σ lies entirely in X_{free}.
- Steer: Given two configurations x_s, $x_t \in X$, the function returns the line segment connecting x_s to x_t.

3.2. P-RRT*

In this section, we show Potential Function Based-RRT* (P-RRT*), which is an extension of RRT*. The RRT* algorithm here is not the original version, but a slightly modified one. The objective of the modification is to reduce the number of calls to the CollisionFree procedure (Perez et al., 2011). Besides using the modified version of RRT*, the outstanding feature of P-RRT* is its incorporation of artificial potential field (APF) into RRT*. The pseudocode for the P-RRT* algorithm is presented in Algorithm 4. Some new

Algorithm 4 P-RRT*.

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;
for $i = 1$ to n **do**
  $x_{rand} \leftarrow SampleFree(i)$;
  $x_{prand} \leftarrow RGD(x_{rand})$;
  $X_{near} \leftarrow Near(V, x_{prand}, min\{\gamma_{RRT*}(\log(card(V)))/card(V)^{1/d}, \eta\})$;
  if $X_{near} = \emptyset$ **then**
    $X_{near} \leftarrow Nearest(V, x_{prand})$
  end if
  $L_{near} \leftarrow GetTuple(x_{prand}, X_{near})$
  $x_{parent} \leftarrow FindBestParent(L_{near}, x_{prand})$
  if $x_{parent} \neq \emptyset$ **then**
    $V \leftarrow V \cup \{x_{prand}\}$;
    $E \leftarrow E \cup (x_{parent}, x_{prand})$;
    $G \leftarrow Rewire(G, x_{prand}, L_{near})$;
  end if
end for
return $G = (V, E)$;

procedures used in P-RRT* are presented below.

- GetTuple: Given x_{prand} and X_{near}, returns L_{near}, which is a sorted data structure in ascending order according to their costs. Each element includes three data attributes, cost, x_{near} and local path σ_{local}.
- FindBestParent: Given L_{near} and x_{prand}, it returns $x_{parent} \in X_{near}$, which is the best parent of x_{prand}.

The RGD procedure is presented in Algorithm 5 in detail. The random sample x_{rand} is adjusted under the attractive force of the

Algorithm 5 RGD(x_{rand}).

$x_{prand} \leftarrow x_{rand}$;
for $n = 1$ to k **do**
  $\vec{F}_{att} \leftarrow (x_{goal} - x_{prand})$;
  $d_{\min} \leftarrow NearestObstacle(X_{obs}, x_{prand})$;
  if $d_{\min} \leq d^*_{obs}$ **then**
    return x_{prand};
  else
    $x_{prand} \leftarrow x_{prand} + \lambda \frac{\vec{F}_{att}}{|\vec{F}_{att}|}$;
  end if
end for
return x_{prand};

goal region X_{goal}, obtaining an improved sample x_{prand}. The NearestObstacle procedure computes the shortest distance from x_{prand} to the obstacle space X_{obs}. There are three parameters k, λ and d^*_{obs} used in RGD. Here, k is a limited number of iterations and the constant d^*_{obs} is a very small distance from X_{obs}, and λ is a small incremental step towards the target. Obviously, these parameters need to be adjusted, but the parameter tuning is omitted here. In this paper, for P-RRT* and PQ-RRT* $\lambda = d^*_{obs} = 0.1$, while $k = 80$.

3.3. Quick-RRT*

Quick-RRT*, as shown in Algorithm 6, has two special adjustments. They are the ChooseParent and the Rewire procedures. In the ChooseParent procedure associated with Quick-RRT*, the search range of the possible parent vertices of x_{rand} includes not only X_{near} but also the ancestry of X_{near} up to a predefined depth. Obviously, expanding the search scope can better optimise the generated tree. However, expanding the search scope does not increase the computation time significantly, for this element of X_{near} usually shares a common parent. Like the ChooseParent procedure, the Rewire procedure also considers the ancestry of the vertex x_{rand}. The Rewire procedure of Quick-RRT* is presented in Algorithm 7.

Note that there is a new procedure Ancestry. Related functions of this procedure are described below.

- ancestor: Given a graph $G = (V, E)$, a vertex p and a constant number $d \in \mathbb{N}$, it returns the $d-th$ parent of p.
- Ancestry: Given a graph $G = (V, E)$ and a vertex p, if the depth d is 0, returns $\emptyset$, otherwise returns $\bigcup_{i=1}^{d} ancestor(G, p, i)$.

Algorithm 6 Quick-RRT*).

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;
for $i = 1$ to n **do**
  $x_{rand} \leftarrow SampleFree(i)$;
  $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
  $\sigma \leftarrow Steer(x_{nearest}, x_{rand})$;
  if $CollisionFree(\sigma)$ **then**
    $X_{near} \leftarrow Near(V, x_{rand}, min\{\gamma_{RRT*}(\log(card(V)))/card(V)^{1/d}, \eta\})$;
    $X_{parent} \leftarrow Ancestry(G, X_{near})$;
    $(x_{parent}, \sigma_{parent}) \leftarrow ChooseParent(X_{near} \cup X_{parent}, x_{nearest}, \sigma)$;
    $V \leftarrow V \cup \{x_{rand}\}$;
    $E \leftarrow E \cup (x_{parent}, x_{rand})$;
    G $\leftarrow$ Rewire-Q-RRT*(G,x_{rand}, X_{near});
  end if
end for
return $G = (V, E)$;

Algorithm 7 Rewire-Q-RRT*(G,x_{rand}, X_{near}).

for each $x_{near} \in X_{near}$ **do**
  for each $x_{from} \in \{x_{rand}\} \cup ancestor(G, x_{rand})$ **do**
    $\sigma \leftarrow Steer(x_{from}, x_{near})$;
    if $\text{Cost}(x_{from} + Cost(\sigma) < Cost(x_near))$ **then**
      if $CollisionFree(\sigma)$ **then**
        $x_{parent} \leftarrow Parent(x_{near})$;
        $E \leftarrow E \backslash \{(x_{parent}, x_{near})\} \cup \{(x_{from}, x_{near})\}$;
      end if
    end if
  end for
end for
return $G = (V, E)$;

4. PQ-RRT*

This section presents the details of the proposed PQ-RRT* algorithm. In order to further accelerate the convergence rate, this paper proposes the PQ-RRT* algorithm based on Quick-RRT* and P-RRT*. Algorithm 8 shows the pseudocode for the PQ-RRT* algorithm.

PQ-RRT* has three main improvements when compared with RRT*. The first point is to change the sampling strategy by adopting the attractive force of the target region. The second point is expanding the search scope of the ChooseParent procedure. The search scope contains not only the neighbor X_{near} but also the ancestry of the X_{near}. The last one is the improvement of the Rewire-PQ-RRT* procedure, which is similar to the second point.

5. Analysis

This section presents the completeness, asymptotic optimality, fast convergence rate and computational complexity of the proposed algorithm. Some symbols used in the analysis are described below. Let ALG denote the label of the algorithms mentioned in the following. Let V_n^{ALG} and E_n^{ALG} be the vertices and edges in the graph G_n^{ALG} generated by an algorithm after n iterations.

5.1. Probabilistic completeness

Problem 1 is considered in this section. It is widely recognised that most sampling-based algorithms can guarantee probabilistic completeness. The concept of probabilistic completeness is formalised below.

Algorithm 8 PQ-RRT*.

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;
for $i = 1$ to n **do**
  $x_{rand} \leftarrow SampleFree(i)$;
  $x_{prand} \leftarrow RGD(x_{rand})$;
  $x_{nearest} \leftarrow Nearest(V, x_{prand})$;
  $\sigma \leftarrow steer(x_{nearest}, x_{prand})$;
  if $CollisionFree(\sigma)$ **then**
    $X_{near} \leftarrow Near(V, x_{prand}, min\{\gamma_{RRT*}(\log(card(V)))/card(V)^{1/d}, \eta\})$;
    $X_{parent} \leftarrow Ancestry(G, X_{near})$;
    $(x_{parent}, \sigma_{parent}) \leftarrow ChooseParent(X_{near} \cup X_{parent}, x_{nearest}, \sigma)$;
    $V \leftarrow V \cup \{x_{rand}\}$;
    $E \leftarrow E \cup (x_{parent}, x_{rand})$;
    G $\leftarrow$ Rewire-PQ-RRT*(G,x_{rand}, X_{near});
  end if
end for
return $G = (V, E)$;

Definition 1 (Probabilistic completeness)**.** Given a triplet $\{x_{init}, X_{obs}, X_{goal}\}$, an algorithm ALG is said to be probabilistically complete if for any robustly feasible path planning problems, $\lim_{n\to\infty}\mathbb{P}(V_n^{ALG}\cap X_{goal}\neq\emptyset)=1$ and the graph returned by ALG includes a path connecting the root x_{init} to $x_{goal}\in X_{goal}$.

Karaman et al. have proved that RRT provides probabilistic completeness and RRT* inherits this property of RRT (Karaman & Frazzoli, 2011). As a result, the probabilistic completeness of P-RRT* and Quick-RRT* algorithms has been deduced (Jeong et al., 2019; Qureshi & Ayaz, 2016). The probabilistic completeness of the proposed algorithm PQ-RRT* is stated in Theorem 1.

Theorem 1 (Probabilistic completeness of PQ-RRT*)**.** *For any given robustly feasible path planning problems*

$\{x_{init}, X_{obs}, X_{goal}\}$, the probability of finding a feasible solution approaches one, i.e.,

$$\lim_{n\to\infty}\mathbb{P}\Big(\exists x_{goal}\in V_n^{PQ-RRT*}\cap X_{goal}\quad such\quad that\quad x_{init}\quad is\quad connected\quad to\quad x_{goal}\in X_{goal}\Big)=1.$$

Proof of Theorem 1. First, we select the same samples as P-RRT*. Moreover, Quick-RRT* just changes the growth trend of the tree but not the connectivity of the tree. Hence the PQ-RRT* provides probabilistic completeness as P-RRT*. □

5.2. Asymptotic optimality

RRT* was proposed by Karaman et al., who proved that RRT does not possess the property of asymptotic optimality that is possessed by RRT*. The related definitions and concepts required to prove the asymptotic optimality of PQ-RRT* are given in the following words.

Let η be a non-negative real number. For a state $y\in X_{free}$, $B_{y,\eta}$ denotes a closed ball region of radius η centered at y. $X_{int_\eta}:=\{y\in X_{free}|B_{y,\eta}\subseteq X_{free}\}$ and $X_{ext_\eta}:=X_{free}\backslash X_{int_\eta}$.

Definition 2 (strong η - clearance)**.** A path is said to have strong η - clearance if and only if all points of the path belong to X_{int_η}.

Definition 3 (weak η - clearance)**.** A path σ_1: [0, 1] is said to have weak η - clearance if there exists a path σ_2: [0, 1] and function ϕ: [0, 1] such that $\phi(0)=\sigma_1$, $\phi(1)=\sigma_2$ and $\phi(\tau)$ is strong η - clearance for $\tau\in(0,1]$.

Let Y_n^{ALG} be a random variable, which denotes the cost of the minimum-cost solution in the graph returned by ALG after n iterations.

Definition 4 (Asymptotic optimality)**.** For any path planning problems, an algorithm ALG is said to be asymptotically optimal if ALG will return a graph including a minimum-cost solution as the number of samples tends to infinity. It can be expressed clearly by the following formula.

$$\mathbb{P}\Big(\lim_{n\to\infty}\sup Y_n^{ALG}=L^*\Big)=1$$

In the above formula, L^* represents the cost of the optimal solution. In fact, P-RRT* introduces intelligent sampling heuristic into RRT* to direct the random samples (Qureshi & Ayaz, 2016). However, the other procedures are the same as RRT*. Therefore, P-RRT* inherits the property of asymptotic optimality. Similarly, PQ-RRT* can be understood as an optimisation version of P-RRT*, so the proposed PQ-RRT* algorithm also possesses the asymptotic optimality property.

5.3. Fast convergence to optimal solution

The proof of fast convergence requires a definition, which shows a critical property of the optimal solution.

Definition 5 (Optimal path planning)**.** If a collision-free path has weak η - clearance, the path is said to be optimal.

Similar to the proof of asymptotic optimality, the property of fast convergence of PQ-RRT* mainly hinges on the P-RRT* algorithm. The following theorem presents the reason why P-RRT* provides fast convergence.

Theorem 2 (Potential guided sampling heuristic RGD(x))**.** *The RGD(x) heuristic guides the random samples towards the goal region in such a manner so that* $\mathbb{P}\big(x_{prand}\in X_{ext_\eta}\big)>0$.

From Theorem 2 and Definition 5, a conclusion that P-RRT* has the property of fast convergence to optimal solution is obtained. Since Quick-RRT* provides a new optimisation frame, the rest procedures of PQ-RRT* are the same as P-RRT*, it can be concluded that PQ-RRT* inherits the property of fast convergence of P-RRT*.

5.4. Computational complexity

In this section, the computational complexity of the proposed PQ-RRT* algorithm is introduced. Let S_n^{ALG} be the number of procedures executed by ALG after n iterations. Theorem 3 presents the fact that PQ-RRT* has the same computational complexity as P-RRT* and Quick-RRT*.

Theorem 3. *There exist two constants α_1 and α_2 such that*

$$\lim_{n\to\infty}\mathbb{E}\left[\frac{S_n^{PQ-RRT*}}{S_n^{P-RRT*}}\right]\leq\alpha_1,$$

$$\lim_{n\to\infty}\mathbb{E}\left[\frac{S_n^{PQ-RRT*}}{S_n^{Quick-RRT*}}\right]\leq\alpha_2.$$

Proof of Theorem 3. Let's first analyze the first formula. The difference between PQ-RRT* and P-RRT* is the integration of Quick-RRT*. The contributions of Quick-RRT* are the improvements on optimisation procedures, the ChooseParent and the Rewire. Since most neighbor vertices have a common ancestor, Quick-RRT* does not greatly increase the computational complexity. Similarly, for the second formula, RGD procedure is considered undoubtedly. For the RGD procedure, it does not increase the number of the samples, but improves the quality of the sample. Hence the two algorithms have the same asymptotic computational complexity. □

6. Simulation results

In this section, PQ-RRT* is compared with the existing algorithms, P-RRT* and Quick-RRT*, in accordance with four benchmarks: two 2-dimensional environments for P-RRT* and two environments for Quick-RRT*. P-RRT* and Quick-RRT* are used for comparative analysis, since they are typical examples of fast convergence algorithms. Due to the randomness of sampling-based algorithms, each algorithm was run 100 times.

The parameters in the simulation are λ, d_{obs}^*, k and the depth d. For a fair comparison, the simulation parameters are the same for all algorithms. In the ChooseParent procedure, $d=2$, and in the Rewire procedure, $d=1$. Two evaluation indicators are utilized to compare the performance of the three algorithms: 'l_f', the length of the first found solution, and '$T_{5\%}$', the time to find a solution of '$1.05*l_{optimal}$', where $l_{optimal}$ is the length of optimal solution. For the purpose of demonstrating the better performance of the PQ-RRT* algorithm, the same random seed is used in the Sample-Free procedure in each comparison. Note that these algorithms will stop once the approximate optimal solutions are found. The simulations are implemented on an Intel Xeon(R) E3-1240 CPU with 8G of RAM. The simulation platform is Matlab. Four test environments are the same size, $100\cdot 100$. The "Fail" value is the number of

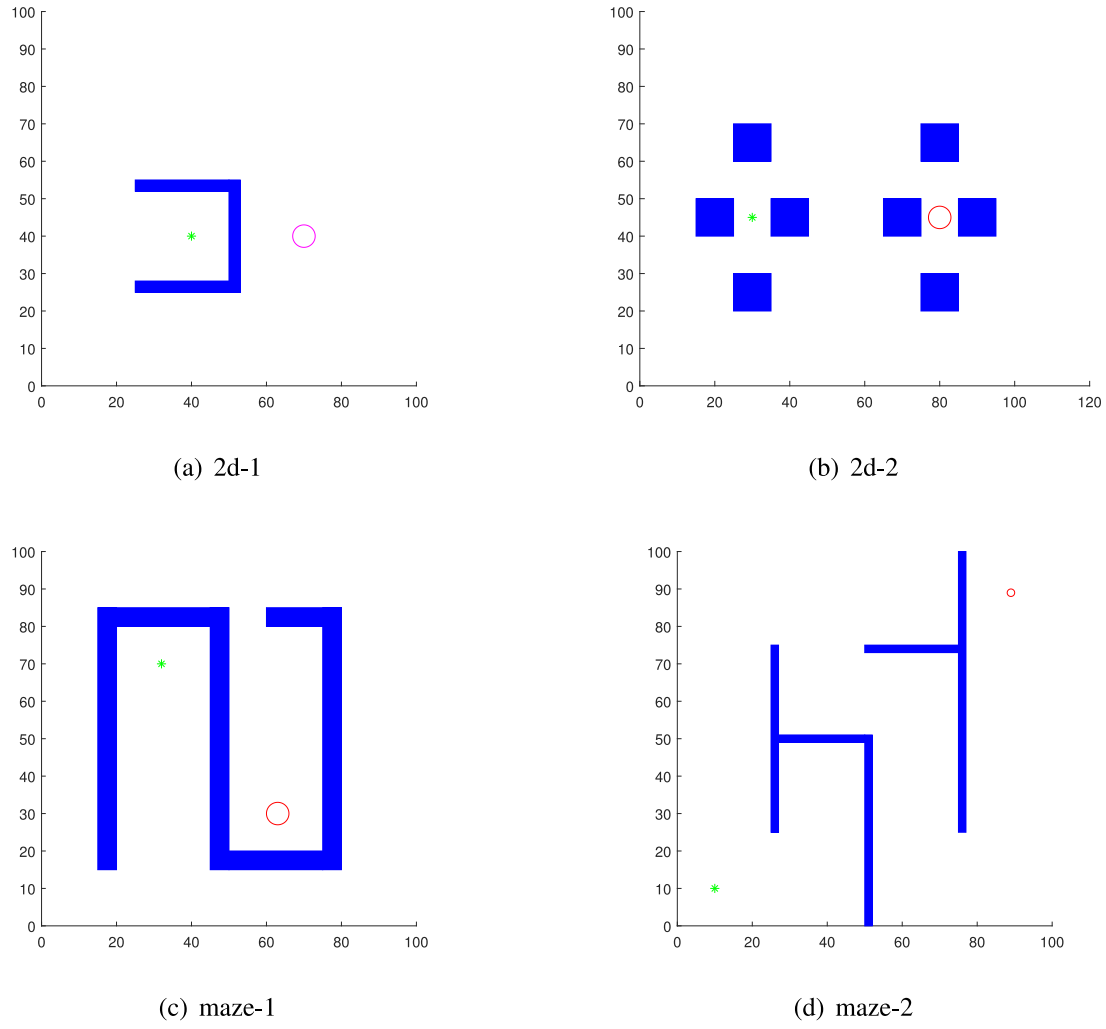


Fig. 1. Four test environments. (a) and (b) are two-dimensional environments for P-RRT*, (c) and (d) are maze environments for Quick-RRT*. (green star: start state, red circle: goal region).

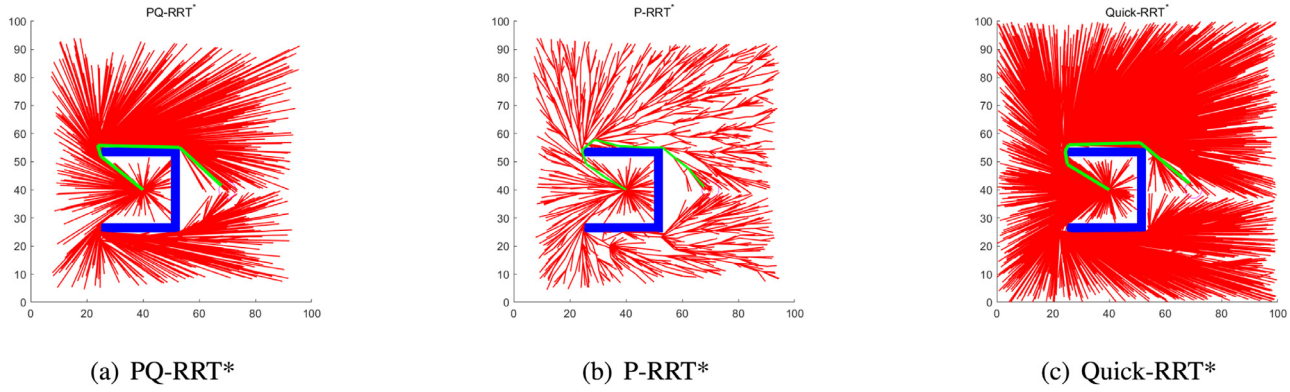


Fig. 2. Performance of the three algorithms in the environment 2d-1.

failures. Failure means an algorithm cannot find the approximate optimal solution within 300 secs. The simulation environments are shown in Fig. 1.

6.1. 2d-1

Environment 2d-1 is shown in Fig. 1(a). Fig. 2 shows the performance of the three algorithms in environment 2d-1. In Fig. 2, the generated paths of PQ-RRT*($l_f = 89.2985$, $T_{5\%} = 1.667$), P-RRT*($l_f = 96.3998$, $T_{5\%} = 6.558$) and Quick-RRT*($l_f = 101.7946$, $T_{5\%} = 32.949$) are shown. It can be seen from the results of this run that the proposed algorithm, PQ-RRT*, has a better initial solution and a faster convergence rate than the other two algorithms. In this environment, $l_{optimal} = 72.8829$.

The statistical results of 100 simulations are described by the boxplots. The unit of the ordinate of the boxplot is seconds. The simulation results of l_f and $T_{5\%}$ are shown in the Figs. 3 and 4 respectively. In the boxplots, p-q denotes PQ-RRT*, p denotes P-

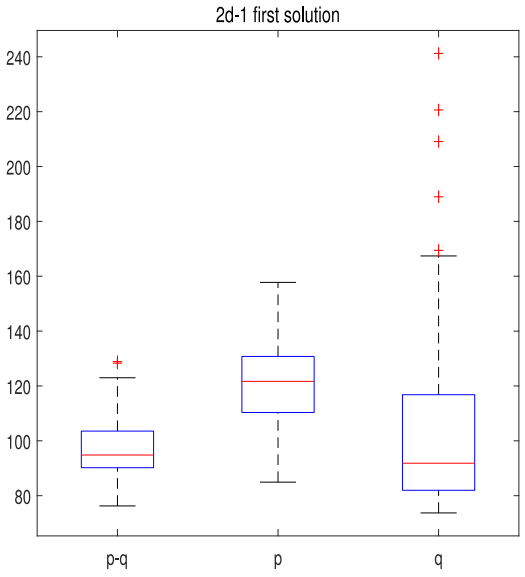


Fig. 3. l_f in the 2d-1.

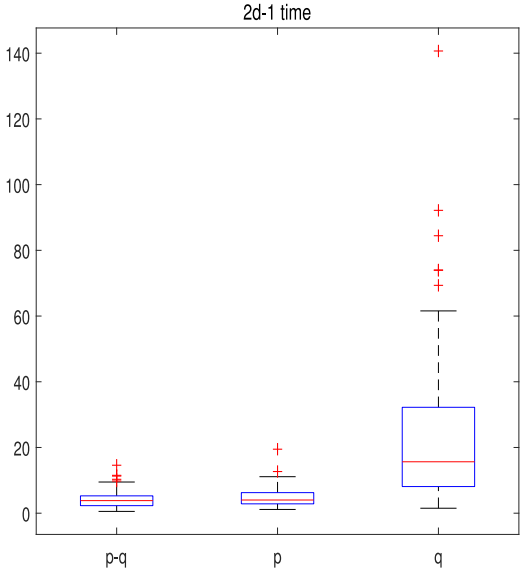


Fig. 4. $T_{5\%}$ in the 2d-1.

Table 1
Comparing algorithms on the quality of the initial solution and convergence rate in 2d-1(Std represents the standard deviation).

Algorithm		Mean	Std	Min	Max	Fail
PQ-RRT*	l_f	96.8742	10.1366	76.2692	128.8751	0
	$T_{5\%}$	4.3554	2.6687	0.547	14.596	
P-RRT*	l_f	120.4426	15.5077	84.9359	157.7636	0
	$T_{5\%}$	4.8352	2.9337	1.14	19.475	
Quick-RRT*	l_f	104.0857	32.7497	73.7135	241.232	0
	$T_{5\%}$	23.7181	22.5681	1.512	140.67	

Table 2
Comparing algorithms on the quality of the initial solution and convergence rate in 2d-2.

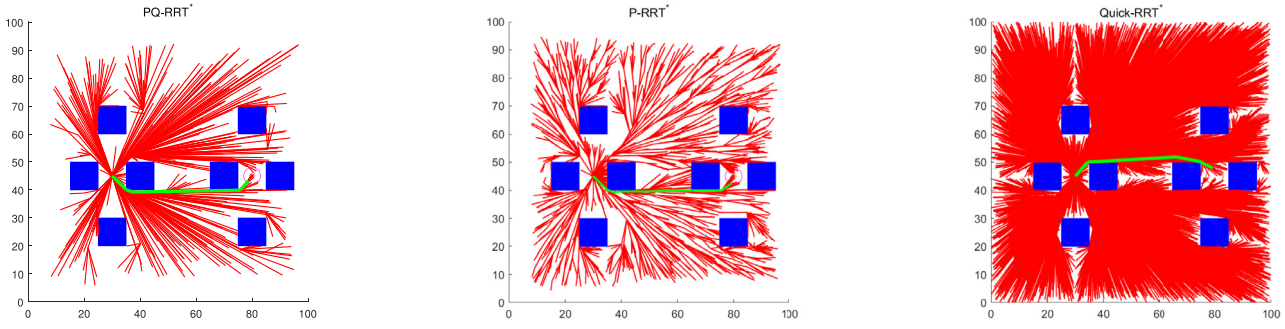
Algorithm		Mean	Std	Min	Max	Fail
PQ-RRT*	l_f	60.3689	3.8795	52.3068	70.9071	0
	$T_{5\%}$	9.2958	7.4132	0.187	40.259	
P-RRT*	l_f	65.6481	7.5297	55.095	105.559	0
	$T_{5\%}$	11.9609	10.6067	0.619	56.207	
Quick-RRT*	l_f	73.4725	21.052	54.1837	144.1125	0
	$T_{5\%}$	40.0247	40.5819	1.51	297.421	

RRT*, and q denotes Quick-RRT*. Although the median of l_f for PQ-RRT* is slightly higher than the median of l_f for Quick-RRT*, PQ-RRT* is better than Quick-RRT* overall. It can be clearly seen from Fig. 4 that in the environment of 2d-1 PQ-RRT* and P-RRT* have a faster convergence rate than Quick-RRT*. The statistics in Table 1 indicate that PQ-RRT* has better stability. Based on the above analysis, PQ-RRT* performs better than P-RRT* and Quick-RRT* in 2d-1.

6.2. 2d-2

Environment 2d-2 is shown in Fig. 1(b). Fig. 5 shows the performance of the three algorithms. In Fig. 5, the generated paths of PQ-RRT*(l_f = 56.1574, $T_{5\%}$= 0.781), P-RRT*(l_f = 56.5157, $T_5\%$ = 4.064) and Q-RRT*(l_f = 54.598, $T_{5\%}$= 33.031) are shown. From the results of this run, we can draw a conclusion that the proposed algorithm, PQ-RRT*, outperforms the other two algorithms in this comparison. In the environment 2d-2, $l_{optimal}$= 52.0711.

The statistical results of 100 simulations are shown in the boxplots. Figs. 6 and 7 plot the results of l_f and $T_5\%$ respectively. It can be clearly seen from Figs. 6 and 7 and Table 2 that PQ-RRT* is outstanding on the two indicators, l_f and $T_5\%$. Based on the above analysis, PQ-RRT* performs better than P-RRT* and Quick-RRT* in 2d-2.



(a) PQ-RRT* (b) P-RRT* (c) Q-RRT*

Fig. 5. Performance of the three algorithms in the environment 2d-2.

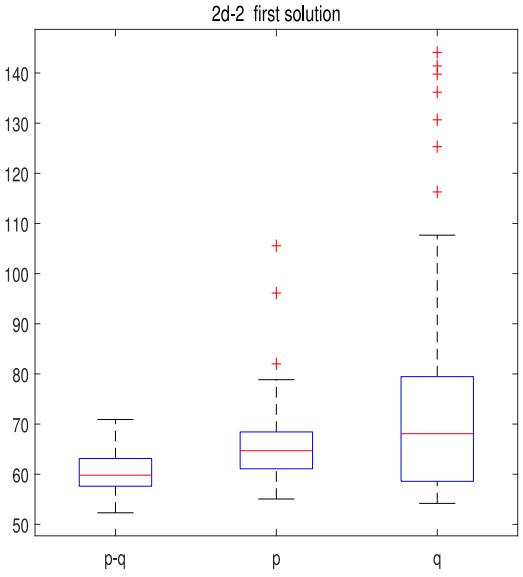


Fig. 6. l_f in the 2d-2.

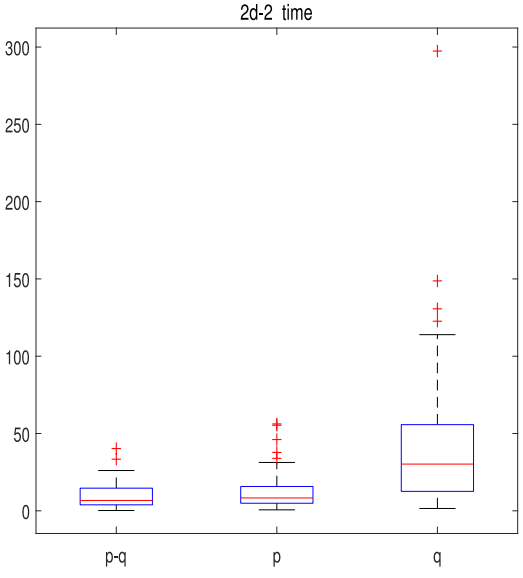


Fig. 7. $T_{5\%}$ in the 2d-2.

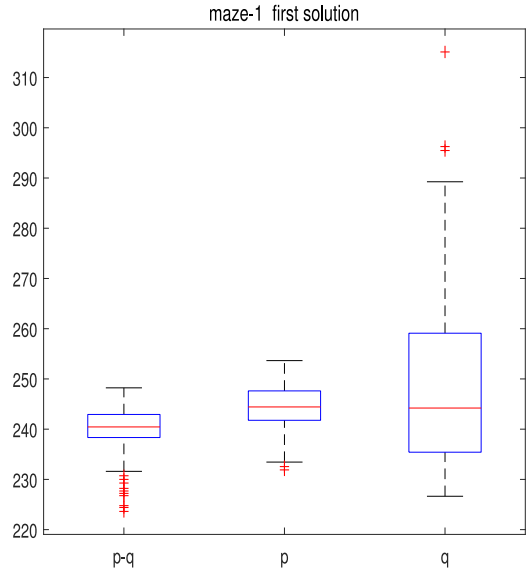


Fig. 9. l_f in the maze-1.

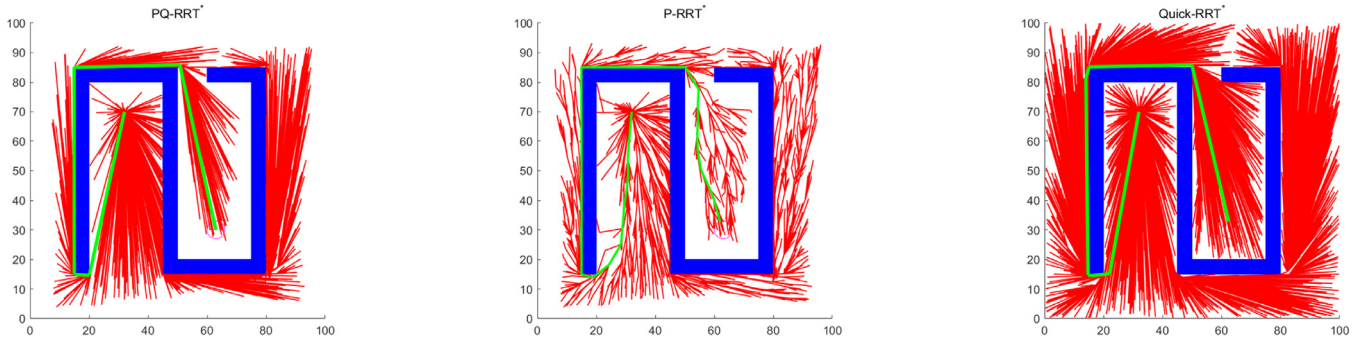
Table 3
Comparing algorithms on the quality of the initial solution and convergence rate in maze-1.

Algorithm		Mean	Std	Min	Max	Fail
PQ-RRT*	l_f	239.4538	5.3222	223.6241	248.227	0
	$T_{5\%}$	9.5649	4.32	2.818	22.464	
P-RRT*	l_f	244.1686	4.482	231.8931	253.6486	4
	$T_{5\%}$	64.2541	65.9887	4.67	280.973	
Quick-RRT*	l_f	250.4689	19.7013	226.6428	315.0957	0
	$T_{5\%}$	23.6302	23.3234	2.768	162.794	

6.3. maze-1

Environment maze-1 is shown in Fig. 1(c). Fig. 8 shows the performance of the three algorithms. In Fig. 8, the generated paths of PQ-RRT*(l_f= 243.824, $T_{5\%}$= 2.849), P-RRT*(l_f = 247.7364, $T_{5\%}$= 4.67) and Q-RRT*(l_f = 252.1599, $T_{5\%}$= 4.723) are shown. As can be seen in Fig. 8, PQ-RRT* can converge to the near optimal solution more quickly and has a lower cost initial solution. In the environment maze-1, $l_{optimal}$= 219.8094. PQ-RRT* requires fewer samples and less time to obtain the near optimal solution.

The statistical results of 100 simulations are shown in the box-plots. Figs. 9 and 10 depict the performance of the algorithms. It can be seen from the box-plots of Figs. 9 and 10 and Table 3



(a) PQ-RRT* (b) P-RRT* (c) Q-RRT*

Fig. 8. Performance of the three algorithms in the environment maze-1.

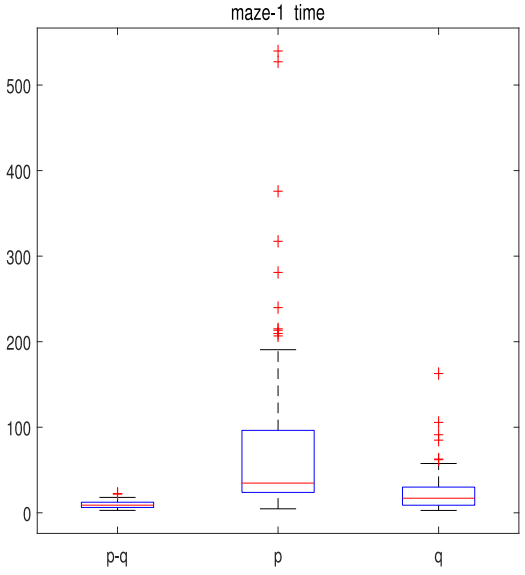


Fig. 10. $T_{5\%}$ in the maze-1.

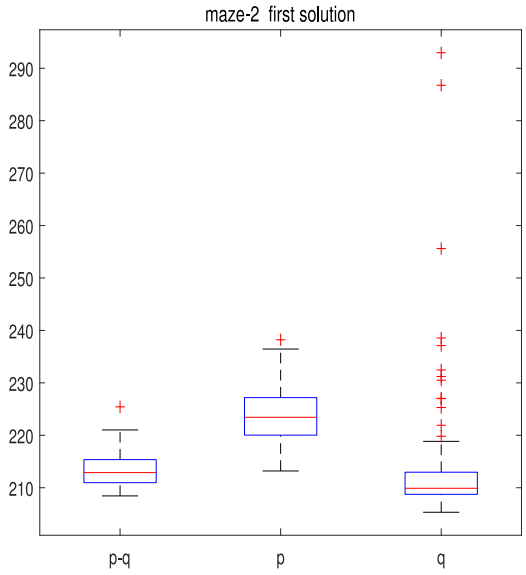


Fig. 12. l_f in the maze-2.

that PQ-RRT* is superior to P-RRT* and Quick-RRT*. It should be pointed out that the failed experiments are not included in the calculation of the mean, standard deviation, maximum and minimum. From the above analysis, it can be concluded that PQ-RRT* performs better than P-RRT* and Quick-RRT* in maze-1.

6.4. maze-2

Environment maze-2 is shown in Fig. 1(d). Fig. 11 shows the performance of the three algorithms. In Fig. 11, the generated paths of PQ-RRT*(l_f= 209.2523, $T_{5\%}$= 2.346), P-RRT*(l_f = 216.7866, $T_{5\%}$= 14.363) and Q-RRT*(l_f = 221.9119, $T_{5\%}$= 13.921) are shown. Fig. 11 demonstrates that the proposed algorithm PQ-RRT* obtains better performance. In the environment maze-2, $l_{optimal}$= 204.1923.

The statistical results of 100 simulations are shown in the boxplots. Figs. 12 and 13 show the comparison of the two indicators, l_f and $T_{5\%}$, respectively. In Fig. 12, the median of PQ-RRT* is slightly higher than that of Quick-RRT*, but Quick-RRT* has many abnormalities that getting the initial solutions takes a long time. In Fig. 13, PQ-RRT* has a faster convergence rate than the other two algorithms. In Table 4, the last column demonstrates PQ-RRT* performs best. Combining two performance indicators, PQ-RRT* performs best in maze-2.

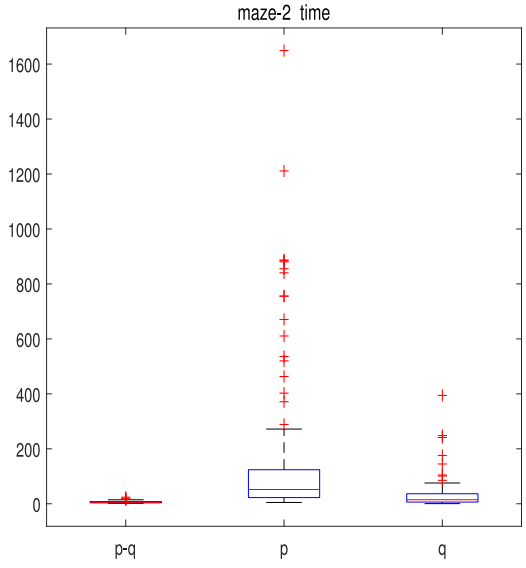
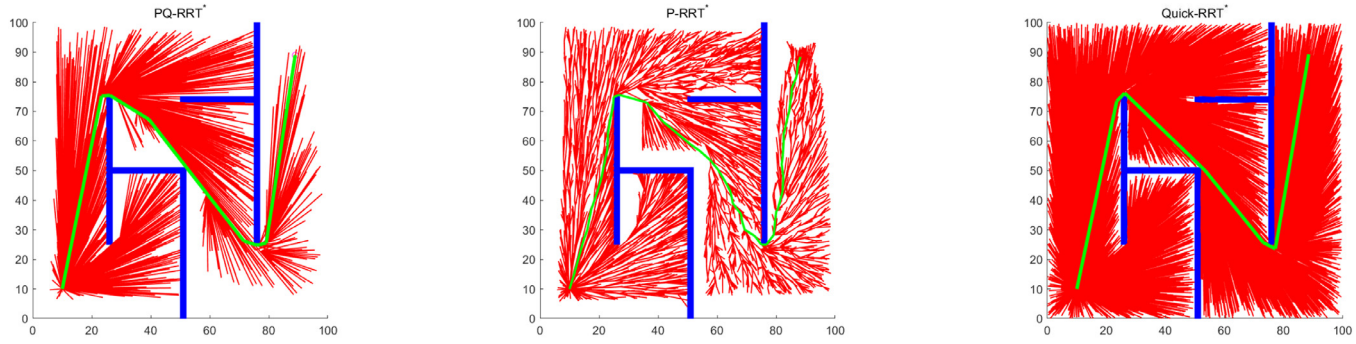


Fig. 13. $T_{5\%}$ in the maze-2.



(a) PQ-RRT* (b) P-RRT* (c) Q-RRT*

Fig. 11. Performance of the three algorithms in the environment maze-2.

Table 4
Comparing algorithms on the quality of the initial solution and convergence rate in maze-2.

Algorithm		Mean	Std	Min	Max	Fail
PQ-RRT*	l_f	213.3854	3.1562	208.4424	225.4243	0
	$T_{5\%}$	6.2253	4.2351	1.583	23.938	
P-RRT*	l_f	223.8193	5.0614	213.1984	238.2166	16
	$T_{5\%}$	62.1314	64.2663	4.722	288.623	
Quick-RRT*	l_f	214.1641	13.4942	205.3227	292.9553	1
	$T_{5\%}$	30.1505	43.3519	0.493	248.213	

The performance of the navigation is usually subject to the initial solution of path planning since the robot will follow the initial solution at the beginning. The boxplots show that the proposed algorithm can generate a relatively better solution. A better initial solution means more energy saving under the same conditions. Moreover, the near optimal solution obtained by PQ-RRT* requires fewer nodes which make memory utilization more efficient. The quality of a path planning algorithm depends not only on asymptotic optimality but also on the convergence rate. The boxplots of the indicator $T_{5\%}$ show that PQ-RRT* has a fast convergence to an optimal solution compared with P-RRT* and Quick-RRT*. From comparisons with P-RRT* and Quick-RRT*, we draw a conclusion that PQ-RRT* performs best overall. PQ-RRT* generates a better initial solution and a fast convergence rate.

7. Conclusion

There has been a great deal of recent research on sampling-based path planning algorithms. RRT* is an optimal algorithm, but it has a slow rate of convergence. In order to address this problem, this paper proposed an improved RRT* algorithm, PQ-RRT*. This paper proves that the proposed algorithm is complete, asymptotically optimal and provides fast convergence to optimal solution. Moreover, PQ-RRT* has the same computational complexity.

Although PQ-RRT* is a promising algorithm, it also has its limitations. When applying PQ-RRT* to mobile robots, the kinematic constraints of the robot must be considered. Another point is that the parameters in the proposed algorithm may not perform best in other complex environments. Therefore, in our future proceedings, we will consider the kinematic constraints and explore an adaptive parameter method for the proposed algorithm. The flexibility of the robot arm can carry out more complex tasks, so applying PQ-RRT* to the robot arm is a potential direction. Static path planning is mostly studied at the present stage, but the actual environments are dynamic, so it is necessary to study the performance of PQ-RRT* in dynamic path planning.

Funding

This work was supported by the National Natural Science Foundation of China [grant number 61573148, 61603358]; the Science and Technology Planning Project of Guangdong Province [grant numbers 2015B010919007, 2016A040403012, 180917144960530]; the Science and Technology Planning Project of Guangzhou [grant number 201604016014]; and the Department of Education of Guangdong Province [grant number 2017KZDXM032].

Declaration of Competing Interest

Authors declare that they have no conflict of interest.

Credit authorship contribution statement

Yanjie Li: Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing. **Wu Wei:** Conceptualization, Methodology, Writing - review & editing. **Yong Gao:** Software, Validation. **Dongliang Wang:** Validation, Writing - original draft. **Zhun Fan:** Writing - original draft.

References

Adiyatov, O., & Varol, H. A. (2013). Rapidly-exploring random tree based memory efficient motion planning. In *2013 IEEE international conference on mechatronics and automation* (pp. 354–359). IEEE.
Akgun, B., & Stilman, M. (2011). Sampling heuristics for optimal motion planning in high dimensions. In *2011 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2640–2645). IEEE.
Alexopoulos, C., & Griffin, P. M. (1992). Path planning for a mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics, 22*(2), 318–322.
Amato, N. M., & Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. In *Proceedings of IEEE international conference on robotics and automation: 1* (pp. 113–120). IEEE.
Asano, T., Asano, T., Guibas, L., Hershberger, J., & Imai, H. (1985). Visibility-polygon search and euclidean shortest paths. In *26th annual symposium on foundations of computer science (SFCS 1985)* (pp. 155–164). IEEE.
Atanacio-Jiménez, G., González-Barbosa, J.-J., Hurtado-Ramos, J. B., Ornelas-Rodríguez, F. J., Jiménez-Hernández, H., García-Ramirez, T., et al. (2011). Lidar velodyne hdl-64e calibration using pattern planes. *International Journal of Advanced Robotic Systems, 8*(5), 59.
Beyer, T., Jazdi, N., Göhner, P., & Yousefifar, R. (2015). Knowledge-based planning and adaptation of industrial automation systems. In *2015 IEEE 20th conference on emerging technologies & factory automation (ETFA)* (pp. 1–4). IEEE.
Brooks, R. A., & Lozano-Perez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, (2), 224–233.
Ferguson, D., & Stentz, A. (2006). Anytime RRTs. In *2006 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5369–5375). IEEE.
Gammell, J. D., Srinivasa, S. S., & Barfoot, T. D. (2014). Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2997–3004). IEEE.
González, D., Pérez, J., Milanés, V., & Nashashibi, F. (2015). A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems, 17*(4), 1135–1145.
Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics, 4*(2), 100–107.
Huang, Y., Li, Z., Jiang, Y., & Cheng, L. (2019). Cooperative path planning for multiple mobile robots via hafsa and an expansion logic strategy. *Applied Sciences, 9*(4), 672.
Islam, F., Nasir, J., Malik, U., Ayaz, Y., & Hasan, O. (2012). RRT*-Smart: Rapid convergence implementation of RRT towards optimal solution. In *2012 IEEE international conference on mechatronics and automation* (pp. 1651–1656). IEEE.
Jeong, I.-B., Lee, S.-J., & Kim, J.-H. (2019). Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. *Expert Systems with Applications, 123*, 82–90.
Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research, 30*(7), 846–894.
Kavraki, L., & Latombe, J.-C. (1994). Randomized preprocessing of configuration space for path planning: Articulated robots. In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS'94): 3* (pp. 1764–1771). IEEE.
Kavraki, L. E., Svestka, P., Latombe, J.-C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation, 12*(4), 566–580.
Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles* (pp. 396–404). Springer.
Koenig, S., Likhachev, M., & Furcy, D. (2004). Lifelong planning A*. *Artificial Intelligence, 155*(1–2), 93–146.
Koren, Y., & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE international conference on robotics and automation* (pp. 1398–1404). IEEE.
Kuffner, J. J., & LaValle, S. M. (2000). RRT-connect: An efficient approach to single–query path planning. In *Proceedings 2000 ICRA. millennium conference. ieee international conference on robotics and automation. symposia proceedings (cat. no. 00ch37065): 2* (pp. 995–1001). IEEE.
LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.
Lee, H.-Y., Shin, H., & Chae, J. (2018). Path planning for mobile agents using a genetic algorithm with a direction guided factor. *Electronics, 7*(10), 212.
Liu, Y., & Badler, N. I. (2003). Real-time reach planning for animated characters using hardware acceleration. In *Proceedings 11th IEEE international workshop on program comprehension* (pp. 86–93). IEEE.
Maekawa, T., Noda, T., Tamura, S., Ozaki, T., & Machida, K.-i. (2010). Curvature continuous path generation for autonomous vehicle using b-spline curves. *Computer-Aided Design, 42*(4), 350–359.
Majeed, A., & Lee, S. (2019). A new coverage flight path planning algorithm based on footprint sweep fitting for unmanned aerial vehicle navigation in urban environments. *Applied Sciences, 9*(7), 1470.

Pandini, M. M., Spacek, A. D., Neto, J. M., & Junior, O. H. A. (2017). Design of a didatic workbench of industrial automation systems for engineering education. *IEEE Latin America Transactions, 15*(8), 1384–1391.

Perez, A., Karaman, S., Shkolnik, A., Frazzoli, E., Teller, S., & Walter, M. R. (2011). Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. In *2011 IEEE/RSJ international conference on intelligent robots and systems* (pp. 4307–4313). IEEE.

Qureshi, A. H., & Ayaz, Y. (2016). Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots, 40*(6), 1079–1093.

Stentz, A. (1997). Optimal and efficient path planning for partially known environments. In *Intelligent unmanned ground vehicles* (pp. 203–220). Springer.

Taylor, R. H., Menciassi, A., Fichtinger, G., Fiorini, P., & Dario, P. (2016). Medical robotics and computer-integrated surgery. In *Springer handbook of robotics* (pp. 1657–1684). Springer.

Valencia-Garcia, R., Martinez-Béjar, R., & Gasparetto, A. (2005). An intelligent framework for simulating robot-assisted surgical operations. *Expert Systems with Applications, 28*(3), 425–433.