IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 12, 2019, accepted January 10, 2020, date of publication January 24, 2020, date of current version January 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2969316

Informed RRT*-Connect: An Asymptotically Optimal Single-Query Path Planning Method

REZA MASHAYEKHI[1], MOHD YAMANI IDNA IDRIS[1], (Member, IEEE), MOHAMMAD HOSSEIN ANISI[2], (Senior Member, IEEE), ISMAIL AHMEDY[1], (Member, IEEE), AND IHSAN ALI[1], (Student Member, IEEE)

[1]Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur 50603, Malaysia
[2]School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K.

Corresponding authors: Reza Mashayekhi (rz.mashayekhi@gmail.com) and Mohd Yamani Idna Idris (yamani@um.edu.my)

This work was supported by the University of Malaya under Grant FP055-2019A and Grant GPF003D-2018.

ABSTRACT Rapidly-exploring Random Trees (RRTs) are successful in single-query motion planning problems. The standard version of RRT grows a tree from a start location and stops once it reached the goal configuration. RRT-Connect is the bidirectional version of RRT, which grows two trees simultaneously. These two trees try to establish a connection to stop searching. RRT-Connect finds solutions faster than RRT. Following that, an asymptotically optimal version of RRT-Connect called RRT*-Connect has been introduced. It not only rewires both trees while they are growing, but also it keeps searching the state space for better solutions than the current one. However, it is inefficient and inconsistent to search all over the state space in order to find better solutions than the current one concerning its single-query nature. The better way is to look through states that can provide a better solution. In this paper, we propose Informed RRT*-Connect, which is the informed version of RRT*-Connect that uses direct sampling after the first solution found. Unlike RRT*-Connect, the proposed method checks only the states that can potentially provide better solutions than the current solution. The proposed method benefited from the properties of RRT*-Connect and informed sampling, which offers low-cost solutions with fewer iterations in comparison to RRT*-Connect. Different simulations in OMPL have been carried out to show the significance of Informed RRT*-Connect in comparison with RRT*, Informed RRT*, and RRT*-Connect.

INDEX TERMS Motion planning, path planning, RRT, RRT-Connect, RRT*, RRT*-Connect, informed sampling.

I. INTRODUCTION

Motion planning problems have various applications such as Self-driving cars, Unmanned Aerial Vehicles (UAVs), medical surgery, computational biology, graphics animation and virtual prototyping [1]–[7]. Motion planners are mostly divided into two groups: graph-based and sampling-based methods. Graph-based methods like Dijkstra's algorithms [8] and A* [9] discretize the configuration space first and then search through the states. They are resolution complete methods so they can return the solution if one exists or they return failure if no path exists. However, they cannot be scaled well with the problem size, such as state dimension or problem range [10].

The associate editor coordinating the review of this manuscript and approving it for publication was Emre Koyuncu.

On the other hand, sampling-based methods such as Expansive Space Trees (ESTs) [11], Probabilistic Roadmap (PRMs) [12] and Rapidly-exploring Random Trees (RRTs) [13] use random sampling to avoid constructing a graph of the configuration space. They have shown practically a significant impact upon the high-dimensional state spaces. They are probabilistically complete, which indicates that the planner will return a solution with a sufficient number of iterations if there is any solution.

Among randomized methods, RRTs [13] have shown a significant performance for single-query planning problems. The standard version of RRT grows as an incremental tree rooted in the start configuration over the collision-free portion of the state space. It stops growing the exploring tree once it spotted one vertex in the goal configuration. The dual-tree version of RRT, RRT-Connect [14], is able to find solutions

faster than RRT, especially when the goal location is challenging to reach regarding the presence of tight passages that the planner has to pass through them to find solutions. RRT-Connect is incrementally growing two trees simultaneously, one from the start location, while another from the goal configuration. These two trees try aggressively to find a connection. The planner stops exploring the state space when a connection between its two trees established.

Nevertheless, RRT and RRT-Connect cannot return optimized solutions due to the lack of optimization process. In order to solve this problem, Karaman and Frazzoli introduced RRT* [15], which explores the state space similar to RRT, but it optimizes the tree by rewiring its branches to achieve near-optimal solutions. RRT* keeps searching the state space after the first solution has been found to return better paths. RRT*-Connect [16] combines RRT-Connect with RRT* to have a bidirectional method which returns near-optimal solutions. Like RRT*, RRT*-Connect keeps searching all over the area in order to return a better solution than the current one.

Although RRT* and RRT*-Connect return near-optimal solutions, they look through all states to optimize their paths. However, it is not an efficient way to decrease the path cost. In [10], [17], Gammell *et al.* demonstrated that the effectiveness of the existing approaches diminishes factorially with the dimension of the configuration space. Therefore, they introduced Informed RRT*, which uses informed sampling on RRT* after a first solution is found. Informed sampling goes through a subset of the configuration space that can provide better solutions. However, Informed RRT* has the problem of other unidirectional tree planners, which is taking time to reach goal configuration, especially when the goal configuration hidden behind narrow passages.

In this paper, we introduce a single-query bidirectional planning method for optimal motion planning problems called Informed RRT*-Connect. Informed RRT*-Connect behaves as RRT*-Connect until a first path is found, after which the proposed method only takes samples from the subset of states that may improve the solution. Like other asymptotically optimal versions of RRTs, Informed RRT*-Connect and RRT*-Connect keep exploring the state space to return near-optimal solutions after the first solution found. However, they are acting differently after a first solution is found. RRT*-Connect look through all over the collision-free part of the state space, while Informed RRT*-Connect search is limited to an ellipsoid subset of the state space which its eccentricity depends on the length of the shortest current solution. Limiting states to a subset gives the ability to the planner to return near-optimal solutions with fewer iterations.

The remainder of the paper is organized as follows. Section II presents the necessary background for the paper, including motion planning definition, informed set and related literature. Section III introduces the proposed method. Section IV presents the simulation and Section V evaluates the simulation's results. Section VI concludes the paper.

II. BACKGROUND

In this section, the necessary background for the paper is presented. It first explains the problem definition. Then, the description of the informed set is presented. Afterwards, all RRT-based methods that are related to this work are explored.

A. PROBLEM DEFINITION

We define the optimal motion planning problem similarly to [10], [15], [16]. Let X be the state space, the states that have collisions with obstacles is named $X_{obs} \subset X$. Complement of X_{obs} is $X_{free} = cl\,(X_{obs})$, all member states of X_{free} are permissible, where $cl\,(.)$ is a closed set. Let $x_{start} \in X_{free}$ be the start location and $X_{goal} \subset X_{free}$ be the goal configuration. A path defined as a set $\sigma\ [0, 1] \rightarrow X_{free}$ such that $\sigma\,(0) = x_{start}$ and $\sigma\,(1) \in X_{goal}$.

Let $c : \Sigma_{X_{free}} \rightarrow \mathbb{R}_{\geq 0}$ be a cost function that assigns a cost value to all collision-free paths. The cost value is the parameter of path optimality. Therefore, the optimal motion planning definition is to search for a path that connects x_{start} to $x_{goal} \in X_{goal}$, while minimizing the cost function. Equation 1 shows the definition of optimized paths.

$$\sigma^* = arg_{\sigma\in\Sigma} min\{c\,(\sigma) \mid \sigma\,(0) = x_{start},\ \sigma\,(1) \in X_{goal},\ \forall s \in [0, 1], \sigma(s) \in X_{free}\} \quad (1)$$

$X_f \subseteq X$ is a subset of the state space which can provide better solution cost than the existing one, c_{best},

$$X_f = \{x \in X \mid f(x) < c_{best}\}. \quad (2)$$

Therefore, planners can increase their convergence rate by limiting their search on states that belong to X_f.

However, $f(\cdot)$ in (2) is unknown, and it is computationally complicated to be found. Instead, a heuristic function, $\hat{f}(\cdot)$, can be considered as an estimation which must not overestimate the actual cost of the path. The definition of $\hat{f}(\cdot)$ is similar to (2).

B. INFORMED SET

The definition of the informed set comes from [10], as a subset of the configuration space that includes states which could improve the optimality of paths. The cost of the path from x_{start} to x_{goal}, $f(x)$, can be divided into two parts. One is the cost of the path from x_{start} to x, $g(x)$, while another is the path cost from x to x_{goal}, $h(x)$. In order to have an estimation of $f(\cdot)$, we need to define the estimation of cost-to-come, $\hat{g}(\cdot)$, and the estimation of cost-to-go, $\hat{h}(\cdot)$.

Euclidean distance is the heuristic for problems that are looking for the minimum length of paths. Therefore, the informed subset that can improve the current solution cost, c_{best}, can be defined as $X_{\hat{f}} = \{x \in X \mid \| x_{start} - x \|_2 + \| x - x_{goal} \|_2 \leq c_{best}\}$, which is the general equation of an n-dimensional prolate hyperspheroid. Fig. 1 shows an ellipse with x_{start} and x_{goal} as its focal points, the transverse diameter is c_{best}, and the conjugate diameter is $\sqrt{c_{best}^2 - c_{min}^2}$, where c_{min} is the Euclidean distance between x_{start} and x_{goal}.

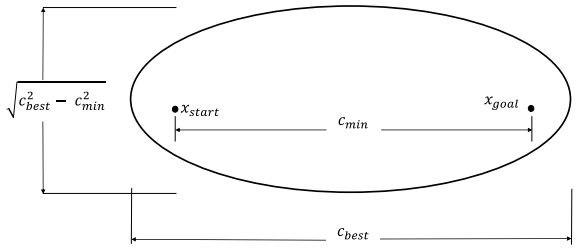


FIGURE 1. The estimated subset, $X_{\hat{f}}$, is an ellipse with x_{start}, and x_{goal}, as its focal points. The ellipse's size is defined based on the minimum cost between x_{start} and x_{goal}, c_{min}, and the cost of the current solution, c_{best}. The ellipse's eccentricity is calculated via c_{min}/c_{best}.

C. RELATED WORK

There are two types of sampling-based motion planners, multi-query, and single-query. Single-query planning is about finding a path between two points in the configuration space which typically named start and goal configurations, while multi-query planning is about making a graph over the collision-free space of the state space in order to connect the different states of the configuration space.

Rapidly-exploring Random Tree (RRT) [13] is a single-query approach that incrementally extends a tree from the start configuration over the collision-free part of the state space and stops exploring once the tree reached the goal configuration. Alg. 1 presents the standard version of RRT in which V, E and G stand for the vertices matrix, the edges matrix, and the tree, respectively. *Sample* function returns a random state. Then the returned sample and the tree will be passed to *Extend* in order to extend the tree toward the x_{rand}.

Algorithm 1 RRT Algorithm

$V \leftarrow \{x_{start}\}$; $E \leftarrow \emptyset$;
$G \leftarrow (V,\ E)$;
for $i\ =\ 1$ *to* n **do**
    $x_{rand} \leftarrow Sample()$;
    $Extend(G = (V, E), x_{rand})$;
end for
return G

Alg. 2 outlines the *Extend* function which first finds the nearest tree vertex to x_{rand}, then implements different constraints via *Steer*. If the connection between $x_{nearest}$ and x_{new} is collision-free, the new vertex and its edge will be added to the tree. This function returns three different outputs due to the status of the new vertex. If x_{rand} is added to the tree, the function output will be *Reached*. If x_{rand} is out of tree reach and another vertex in the direction of x_{rand} but nearer to the tree is added to the tree, the function's output will be *Advanced*. If, due to the presence of an obstacle, x_{rand} cannot be added to the tree, the function output will be *Trapped*.

RRT explores the collision-free part of the state space rapidly so that this simple but efficient method is successful for many practical applications. However, RRT has some problem to sample the goal configuration in some scenarios

Algorithm 2 Extend Function

function $Extend(G = (V, E), x)$
    $x_{nearest} \leftarrow Nearest(G,\ x)$;
    $x_{new} \leftarrow Steer(x_{nearest},\ x)$;
    if $isCollisionFree(x_{nearest},\ x_{new})$ **then**
        $V \leftarrow V \bigcup \{x_{new}\}$;
        $E \leftarrow E \bigcup \{x_{nearest},\ x_{new}\}$;
        if $(x_{new} = x)$ **then**
            return *Reached*;
        else
            return *Advanced*;
        end if
    end if
    return *Trapped*;
end function

in which reaching the goal configuration requires to pass through one or more narrow passages. Moreover, the path returned by RRT is non-optimal [15].

To solve this problem, Kuffner and LaValle proposed RRT-Connect [14], which is a bidirectional version of RRT. RRT-Connect explores the state space by using dual-tree. One of these two trees is rooted in the start configuration, while another is rooted in the goal configuration. These two trees are growing simultaneously and trying to establish a connection. The planner keeps expanding the trees until a connection established. Alg. 3 demonstrates the RRT-Connect approach. It initializes its trees and then expands them over the collision-free part of the state space.

Algorithm 3 RRT-Connect Algorithm

$V_a \leftarrow \{x_{start}\}$; $E_a \leftarrow \emptyset$;
$V_b \leftarrow \{x_{goal}\}$; $E_b \leftarrow \emptyset$;
$G_a \leftarrow (V_a,\ E_a)$; $G_b \leftarrow (V_b,\ E_b)$;
for $i\ =\ 1$ *to* n **do**
    $x_{rand} \leftarrow Sample()$;
    if $Extend(G_a, x_{rand}) \neq Trapped$ **then**
        if $Connect(G_b, x_{new}) = Reached$ **then**
            return $G_a,\ G_b$;
        end if
    end if
    $Swap(G_a,\ G_b)$;
end for
return $G_a,\ G_b$;

In Alg. 3, the planner first extends $Tree_a$ and then takes the newly added vertex, x_{new}, of $Tree_a$ and $Tree_b$ as inputs to *Connect* function. *Connect* function keeps calling *Extend* function in order to link $Tree_b$ and x_{new} of $Tree_a$. *Connect* function will be stopped when the connection is found or an obstacle blocks the connection. If *Connect* function returns *Reached*, it means that the trees are now connected. Therefore, exploring is finished, and the planner will return the trees.

Algorithm 4 Connect Function

function *Connect*$(G,\ x)$
  repeat
    $S \leftarrow Extend(G,\ x)$;
  until $S \neq Advanced$;
  return S;
end function

Although RRT-Connect is taking fewer iterations than RRT to find solutions in many problems, its output remains non-optimal like RRT's.

In 2011, Karaman and Frazzoli [15] introduced RRT*, which is the optimized version of RRT. It guarantees optimal solutions in which the optimality of solutions is defined based on the path length. RRT* keeps rewiring the tree to minimize the cost function.

In other words, RRT* is incrementally rewiring the tree based on the newly added state to the tree. The newly added states are considered as replacement parents for other existing nearby states in the tree. Alg. 5 outlines RRT* procedure, which is similar to RRT, but it calls *Extend** function that is the optimized version of *Extend* function. *Extend** includes the rewiring procedure, which tries to optimize the tree vertices near the newly added vertex. Alg. 6 outlines this procedure.

Algorithm 5 RRT* Algorithm

$V \leftarrow \{x_{start}\}$; $E \leftarrow \emptyset$;
$G \leftarrow (V,\ E)$;
for $i\ =\ 1$ *to* n **do**
  $x_{rand} \leftarrow Sample()$;
  $Extend^*(G,\ x_{rand})$;
end for
return G

RRT* is returning near-optimal solutions. Nonetheless, it still holds the problem of unidirectional searches, such as finding the first solution by consuming time in comparison with bidirectional methods.

In 2015, Klemm *et al.* [16] presented the optimized version of RRT-Connect. They replaced RRT by RRT* in RRT-Connect and made RRT*-Connect, which is maintaining both trees during their growth like RRT*. It is fast to find solutions like RRT-Connect. Moreover, it optimizes its solution while exploring the state space. Alg. 7 presented the RRT*-Connect.

*Connect** function is like *Connect* function, but *Connect** calls *Extend** instead of *Extend* function.

Although RRT*-based methods obtain near-optimal paths, they are not consistent according to their single-query nature, and they become expensive in high dimensions [10]. In order to minimize the path cost, it is better to look through the states that can achieve the less path cost. However, it is computationally expensive to find the states that can provide better solutions than the first one.

Algorithm 6 Extend* Function

function $Extend^*(G = (V, E), x)$
  $x_{nearest} \leftarrow Nearest(G,\ x)$;
  $x_{new} \leftarrow Steer(x_{nearest},\ x)$;
  if $isCollisionFree(x_{nearest},\ x_{new})$ **then**
    $V \leftarrow V \bigcup \{x_{new}\}$;
    $x_{min} \leftarrow x_{nearest}$;
    $X_{near} \leftarrow Near(G,\ x_{new},\ r_{RRT^*})$;
    $c_{min} \leftarrow Cost(x_{nearest},\ G)$
    $+Cost(Line(x_{nearest},\ x_{new}))$;
    for each $x_{near} \in X_{near} \backslash x_{nearest}$ **do**
      if $isCollisionFree(x_{near},\ x_{new})$ &
       $(Cost(x_{near},\ G) + Cost(Line(x_{near},\ x_{new}))$
        $< c_{min})$ **then**
        $x_{min} \leftarrow x_{near}$;
        $c_{min} \leftarrow Cost(x_{near},\ G)$
        $+Cost(Line(x_{near},\ x_{new}))$;
      end if
    end for
    $E \leftarrow E \bigcup \{x_{min},\ x_{new}\}$;
    for each $x_{near} \in X_{near} \backslash x_{min}$ **do**
      if $isCollisionFree(x_{near},\ x_{new})$ &
        $(Cost(x_{new},\ G)+Cost(Line(x_{new},\ x_{near}))$
        $< Cost(x_{near},\ G))$ **then**
        $x_{parent} \leftarrow Parent(x_{near},\ G)$;
        $E \leftarrow E \setminus \{(x_{parent},\ x_{near})\}$;
        $E \leftarrow E \bigcup \{(x_{new},\ x_{near})\}$;
      end if
    end for
    if $(x_{new} = x)$ **then**
      return *Reached*;
    else
      return *Advanced*;
    end if
  end if
  return *Trapped*;
end function

Algorithm 7 RRT*-Connect Algorithm

$V_a \leftarrow \{x_{start}\}$; $E_a \leftarrow \emptyset$;
$V_b \leftarrow \{x_{goal}\}$; $E_b \leftarrow \emptyset$;
$G_a \leftarrow (V_a,\ E_a)$; $G_b \leftarrow (V_b,\ E_b)$;
for $i\ =\ 1$ *to* n **do**
  $x_{rand} \leftarrow Sample()$;
  if $Extend^*(G_a, x_{rand}) \neq Trapped$ **then**
    $Connect^*(G_b, x_{new})$;
  end if
  $Swap(G_a,\ G_b)$;
end for
return $G_a,\ G_b$;

Gammell *et al.* [10], [17] presented an ellipsoid subset of the state space, which is an estimation of all states that can provide better solutions than the existing one. The subset is

Algorithm 8 Connect* Function

```
function Connect*(G, x)
    repeat
        S ← Extend*(G, x);
    until S ≠ Advanced;
    return S;
end function
```

an admissible estimation which is not computationally expensive to achieve. They implemented the ellipsoid subset on RRT* and made Informed RRT*, which uses direct sampling after the first solution is found. Informed RRT* can achieve near-optimal solutions faster than the standard version of RRT*. However, it has the same problem of unidirectional searches, which is finding the first solution in complex configuration spaces, which only offers narrow passages between the start and the goal configurations.

III. THE PROPOSED METHOD

In this section, we present the proposed method, Informed RRT*-Connect, which is an asymptotically optimal dual-tree RRT-based planner. Informed RRT*-Connect acts like RRT*-Connect to explore the configuration space before the first solution is found. Afterwards, it limits the search within an ellipsoidal subset to return better solutions than the current one by fewer iterations in comparison to the standard version of RRT*-Connect.

Algorithm 9 Informed RRT*-Connect Algorithm

```
V_a ← {x_start}; E_a ← ∅;
V_b ← {x_goal}; E_b ← ∅;
G_a ← (V_a, E_a); G_b ← (V_b, E_b);
X_soln ← ∅;
c_best ← ∞;
for i = 1 to n do
    previous_c_best ← c_best;
    c_best ← CalculateShortestPathLengh(X_soln);
    if c_best < previous_c_best then
        PruneTree(V, E, c_best);
    end if
    x_rand ← InformedSample(x_start, x_goal, c_best);
    if Extend*(G_a, x_rand) ≠ Trapped then
        Connect*(G_b, x_new);
    end if
    Swap(G_a, G_b);
    if isSolutionFound(x_new) then
        X_soln ← X_soln ⋃ {x_new}
    end if
end for
return G_a, G_b;
```

Alg. 9 outlines the steps of Informed RRT*-Connect method. It first initializes the algorithm variables such as both trees, then starts iterating both of them. It starts exploring the configuration space similarly to RRT*-Connect until a solution is found. After a solution found, the cost of the shortest path is calculated by *CalculateShortestPathLength* function, and then the returned value of this function will be stored in c_{best}. If c_{best} is getting smaller than its previous amount, the tree will be pruned based on c_{best} value. Then, c_{best} will be passed to *InformedSample* function to limit the search within the informed subset. After taking a sample from the *InformedSample* function, the planner starts to extend $tree_a$ and try to make a connection between the newly added vertex and $tree_b$. Afterwards, the two trees will be swapped for the next iteration. If a new solution is found, it will be added to the solution matrix, X_{soln}.

A. GRAPH PRUNING

Graph pruning is a method that removes some vertices from the tree in order to make it smaller so that the planning process will be carried out faster. Karaman *et al.* [18] implemented a graph pruning technique on a version of RRT*, which is able to improve paths. They calculate the estimated cost of each vertex as the sum of cost-to-come and estimated cost-to-go. If the estimated cost is higher than the shortest path's length, then the vertex must be removed from the tree; otherwise, the vertex will remain in the tree. However, the cost-to-come of the vertices may be getting smaller due to the rewiring process. As a result, this method may remove vertices that could potentially provide better solutions.

Gammell *et al.* [17] uses another heuristic for pruning the tree. The cost-to-come of a vertex estimated as the Euclidean distance between x_{start} and the vertex. Similarly, the cost-to-go estimated as the Euclidean distance between the vertex and x_{goal}. If the estimated cost is smaller than the shortest path length, then the vertex will not be removed from the tree. In other words, this method keeps only the vertices that either itself or one of its children located inside the informed set. The details of this method is presented in Alg. 10.

Algorithm 10 *PruneTree*($V \subseteq X$, $E \subseteq V \times V$, $c_{best} \in \mathbb{R}_{\geqslant 0}$)

```
do
    V_prune ← {v ∈ V | f̂(v) > c_best, and ∀w ∈ V, (v, w) ∉ E};
    E ← E \ {(u, v) ∈ E | v ∈ V_prune};
    V ← V_prune;
while V_prune ≠ ∅;
```

PruneTree function removes the vertices of the tree which are not parent of any other vertices and their estimated cost, $\hat{f}(v) = \hat{g}(v) + \hat{h}(v)$, is greater c_{best}. These vertices are not able to provide better solutions than the existing one. In other words, this function removes the leaves of the tree those have estimated cost more than c_{best}.

B. DIRECT SAMPLING OF AN ELLIPSOIDAL SUBSET

The idea of direct sampling of an ellipsoidal subset comes from [10]. In order to achieve uniformly distributed sampling inside the ellipsoidal subset $X_{ellipse} \sim \mathcal{U}(X_{ellipse})$, we can

transform uniformly distributed samples from unit n-ball to ellipsoidal subset. $X_{ball} \sim \mathcal{U}(X_{ball})$, $x_{ellipse} = Lx_{ball} + x_{center}$, where $X_{center} = (x_{f1} + x_{f2})/2$ is the center of the hyperellipsoid with its two focal points, x_{f1} and x_{f2}, and $X_{ball} = \{x \in X \mid \| x \|_2 \leq 1\}$ [19].

This transformation will be calculated by Cholesky decomposition of the hyperellipsoid matrix, $S \in \mathbb{R}^{n\times n}$, $LL^T \equiv S$, $(x - x_{center})^T S(x - x_{center}) = 1$, S having eigenvectors corresponding to the axes of the hyperellipsoid, $\{a_i\}$, and eigenvalues corresponding to the squares of its radii, $\{r_i^2\}$. The transformation, L, maintains the uniform distribution in $X_{ellipse}$ [20].

Therefore, the transformation of $X_{\hat{f}}$ can be achieved just by transverse the radii and axis. The diagonal matrix of the transverse axis is $S = diag\{\frac{c_{best}^2}{4}, \frac{c_{best}^2 - c_{\min}^2}{4}, \ldots, \frac{c_{best}^2 - c_{\min}^2}{4}\}$ and decomposition $L = diag\{\frac{c_{best}}{2}, \frac{\sqrt{c_{best}^2 - c_{\min}^2}}{2}, \ldots, \frac{\sqrt{c_{best}^2 - c_{\min}^2}}{2}\}$ where $diag\{.\}$ stands for a diagonal matrix.

In order to rotate the hyperellipsoid to the world frame, the Wahba problem [21] can solve it. The rotation matrix is calculated by

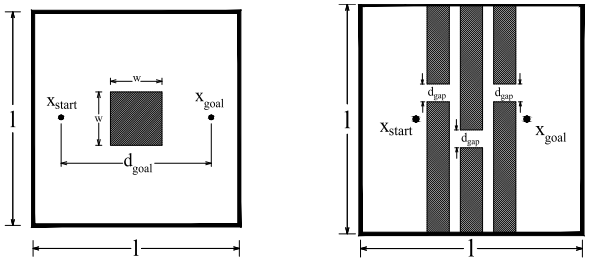
$C = U\ diag\{1, \ldots, 1, det(U)det(V)\}V_T$, where $det(.)$ stands for matrix determinant. $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{n\times n}$ are unitary matrices of $U\Sigma V^T \equiv M$ through singular value decomposition. The matrix M is calculated via the outer product of the first column of the identity matrix, 1_1, and the transverse axis on the world frame, a_1, $M = a_1 1_1^T$, where $a_1 = (x_{goal} - x_{start})/\| x_{goal} - x_{start} \|_2$.

Thus, the below formula will calculate the states that belong to the informed subset. $x_{\hat{f}} = CLx_{ball} + x_{center}$, Alg. 11 presents the informed sampling procedure.

Algorithm 11 *InformedSample*(x_{start}, x_{goal}, c_{max})

```
if c_max < ∞ then
    c_min ← ‖ x_goal − x_start ‖_2;
    x_center ← (x_start + x_goal)/2;
    C ← RotationToWorldFrame(x_start, x_goal);
    r_1 ← c_max/2;
    {r_i}_{i=2,...,n} ← (√(c_max^2 − c_min^2))/2;
    L ← diag{r_1, r_2, ..., r_n};
    x_ball ← SampleUnitBall;
    x_rand ← (CLx_ball + x_center) ⋂ X;
else
    x_rand ~ U(X);
end if
return x_rand;
```

InformedSample function is for sampling the configuration space. If the c_{best} is not infinity, it means that a path between x_{start} and x_{goal} has already been found. Therefore, *InformedSample* must return samples within the ellipsoid subset. If no path is found, the function does not limit the configuration space and returns a sample over the configuration space.



(a) Single Cube (b) Multiple Narrow Passages

FIGURE 2. **Single Cube configuration space, (a), has an obstacle located at the center of the configuration space, and the width of the obstacle is a random variable uniformly distributed over the range [0.25, 0.5]. (b) shows Multiple Narrow Passages configuration space, which only offers planners to pass through three gaps between x_{start} and x_{goal} to produce solutions.**

C. REWIRING NEIGHBORHOOD

RRT*-Connect is rewiring the neighbor vertices of new states so that it almost-surely converges asymptotically to the optimum solution. There are two types of definitions for the neighborhood in a tree structure: r-disc variant and k-nearest variant.

1) r-DISC VARIANT

In this definition, all vertices which are located within a radius, $r^*_{RRT*-Connect}$ will be considered as the neighbors.

$$r^*_{RRT*-Connect} := \left(2\left(1 + \frac{1}{n}\right)\left(\frac{\lambda(X)}{\zeta_n}\right)\left(\frac{\log(|V|)}{|V|}\right)\right)^{\frac{1}{n}} \quad (3)$$

where $|\cdot|$ is the cardinality of a set, the Lebesgue measure of an n-dimensional unit ball and the Lebesgue measure of a set are shown by ζ_n and $\lambda(\cdot)$, respectively.

Informed RRT*-Connect searches the state space to find a solution. Then, the search will be limited to a subset of the configuration space so that the rewiring will be a function of the number of vertices in the informed set, $|V \cap X_{\hat{f}}|$, and its measure,

$$\lambda\left(X_{\hat{f}}\right) \leq \min\{\lambda(X), \lambda(X_{subset})\}.$$

Thus, $r^*_{RRT*-Connect}$ will be updated as

$$r^*_{RRT*-Connect} \leq \left(2\left(1 + \frac{1}{n}\right)\left(\frac{\min\{\lambda(X), \lambda(X_{subset})\}}{\zeta_n}\right) \times \left(\frac{\log\left(\left|V \cap X_{\hat{f}}\right|\right)}{\left|V \cap X_{\hat{f}}\right|}\right)\right)^{\frac{1}{n}} \quad (4)$$

2) k-NEAREST VARIANT

In this definition, the near neighbor set includes k closest vertices to the new vertex.

$$k^*_{RRT*-Connect} := e\left(1 + \frac{1}{n}\right)\log(|V|). \quad (5)$$

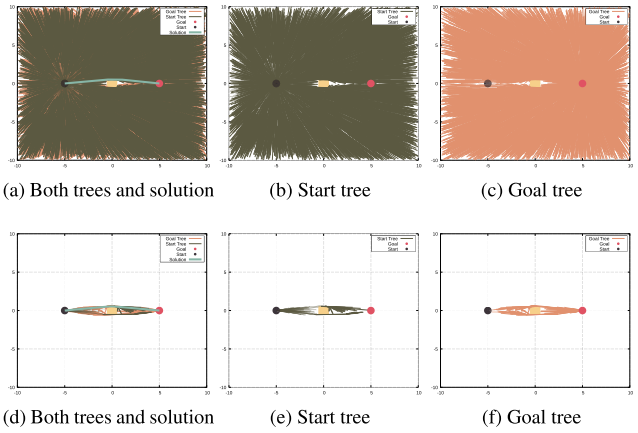


FIGURE 3. One run of RRT*-Connect and Informed RRT*-Connect in the Single Cube scenario shown when the obstacle width is 0.5, and $1/d_{goal} = 2$. (a), (b), and (c) show RRT*-Connect's trees, while (d), (e), and (f) demonstrate the trees obtained from Informed RRT*-Connect.

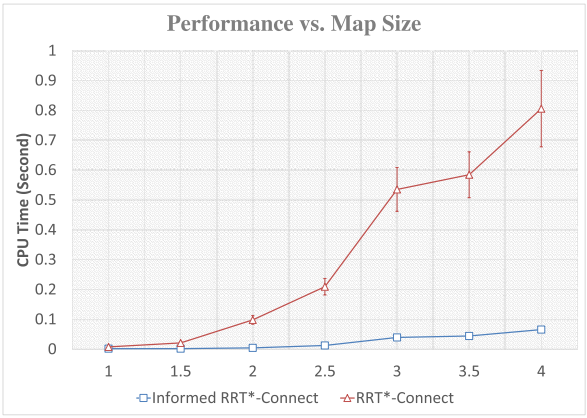


FIGURE 4. The median time required by Informed RRT*-Connect and RRT*-Connect to find solution cost within 2% of the optimal path cost for different amounts of l/d_{goal} in Single Cube scenario obtained from 100 runs. Error bars demonstrate a nonparametric 95% confidence interval for the median time. It can be seen that Informed sampling performs best comparatively when the distance between x_{start} and x_{goal}, d_{goal}, is much shorter than the size of the configuration space, l.

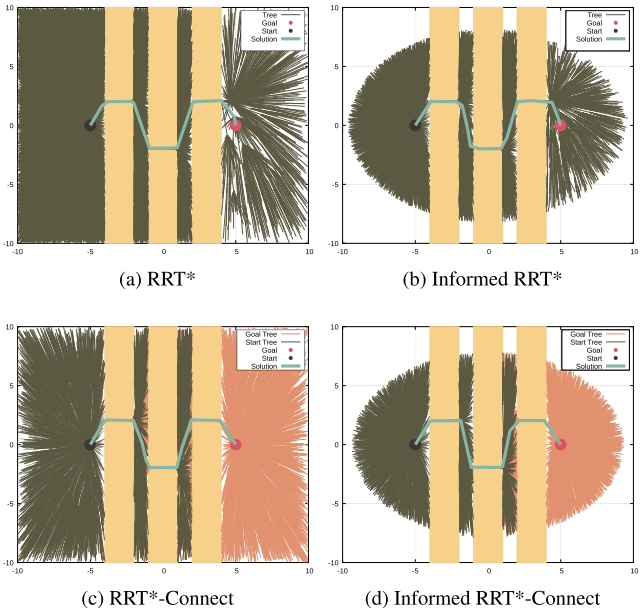


FIGURE 5. One example of the trees and solution paths obtained by the planners in Multiple Narrow Passages scenario.

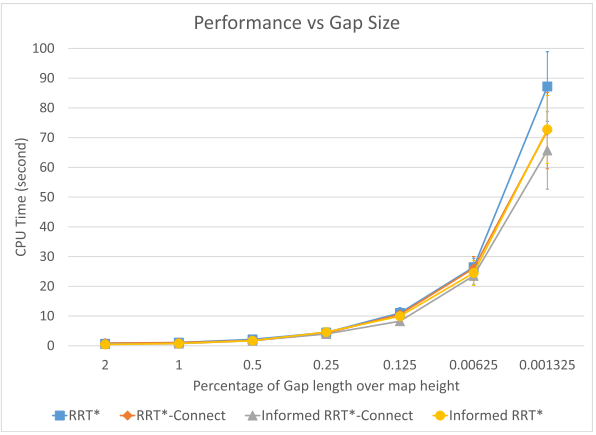


FIGURE 6. The median time required by the planners to find solution cost within 2% of the optimal path cost for different gap sizes in Multiple Narrow Passages scenario obtained from 100 runs. Error bars denote a nonparametric 95% confidence interval for the median time.

Similar to r-disc for Informed RRT*-Connect, the $k^*_{RRT^*-Connect}$ will be updated as

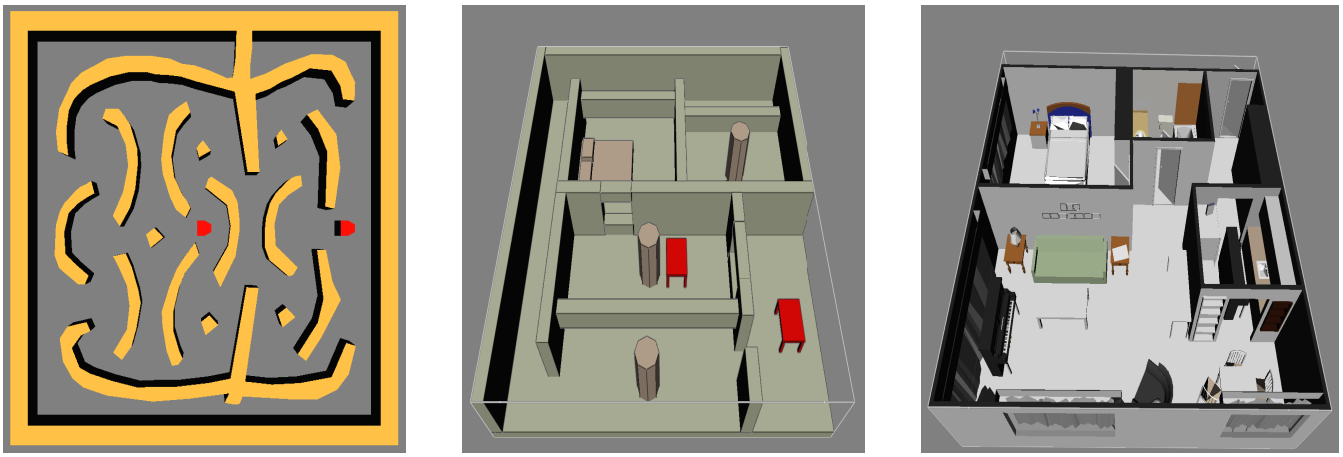
$$k^*_{RRT^*-Connect} := e\left(1+\frac{1}{n}\right)\log\left(\left|V \cap X_{\hat{f}}\right|\right). \qquad (6)$$

If the subset contains less number of vertices in comparison to the whole configuration space, then the rewiring neighborhoods of the Informed RRT*-Connect (4) and (6) will be smaller than (3) and (5). Therefore, the planner requires less computational time for the rewiring process. As a result, its performance will be improved. However, if the distance between x_{start} and x_{goal} be relatively big, then the informed set is not limiting the rewiring neighborhoods considerably so that Informed RRT*-Connect and RRT*-Connect act similarly.

RRT*-Connect is a probabilistic complete and almost-sure asymptotically optimal planner. Informed RRT*-Connect acts exactly like RRT*-Connect for the first solution so that it is probabilistically complete. Moreover, Informed RRT*-Connect maintains a uniform sample distribution over the ellipsoid subset, in which it implements rewiring that satisfies the bound mentioned in [15]. Therefore, it is also almost-surely asymptotically optimal motion planner.

IV. SIMULATION

Informed RRT*-Connect was evaluated on simulated problems in $\mathbb{R}^2$, $\mathbb{R}^3$, and $\mathbb{R}^6$ using Open Motion Planning Library (OMPL) [22]. The simulations carried out on a laptop with the Intel Core i7-6700HQ processor and 12GB of RAM. The laptop operating system was Ubuntu 16.04. Informed RRT*-Connect has been compared with RRT*-Connect,



(a) Maze planar (b) Home (c) Apartment Hard

FIGURE 7. The three OMPL App scenarios which have been tested in the simulation section. (a) is a 3DoFs configuration space, while (b) and (c) are 6DoFs problems.

RRT*, and Informed RRT* on different scenarios in order to demonstrate the significance of the proposed method over the existing planners.

The first two simulations configuration spaces are shown in Fig. 2. These two simulations are simple which designed for 2 Degree of Freedoms (DoFs) problems. The rest of the simulations are the configuration spaces provided by OMPL App. They are shown in Fig. 7. Maze planar configuration space. Fig. 7a, is a 3D problem, while Home, Fig. 7b, and Apartment, Fig. 7c, are 6D problems.

The planners were run 100 times in each scenario in order to get the median of their results.

A. SINGLE CUBE

The idea of this simulation comes from [10], which designed to examine the impact of informed sampling in relation to the size of the map and distance between x_{start} and x_{goal}. This simulation includes an obstacle located in the center of the configuration space, which is a square with a randomly selected width between [0.25, 0.5]. The Single Cube map is shown in Fig. 2a, in which the distance between x_{start} and x_{goal} is shown as d_{goal}, and the length of map is shown as l. The simulation has been carried out for different values of l/d_{goal}. Fig. 3 demonstrates the trees obtained from RRT*-Connect and Informed RRT*-Connect in this scenario.

The results obtained from 100 independent runs in Fig. 4 shows that Informed RRT*-Connect and RRT*-Connect, both found solutions at almost the same time when d_{goal} was equal to l, while at the smallest ratio, 4, Informed RRT*-Connect was about ten times faster than RRT*-Connect.

B. MULTIPLE NARROW PASSAGES

This simulation has been carried out to examine the ability of planners to find paths in the problems that offer only narrow passages. Fig. 2b shows the configuration space of Multiple Narrow Passages, in which there are three barriers in the middle of the configuration space. Each barrier has a slight passage with the height that shows as d_{gap}. All these passages have the same height. In order to solve this problem, the planner has to pass through all the three gaps to connect the x_{start} and x_{goal} configurations together. This simulation has been carried out on the different heights of the passages so as to examine the effect of the passage height on the planners. The gap height in this simulation starts from 2% of the map height, l, and getting smaller until 0.001325% of l. One run of each planner is shown in Fig. 5.

For each gap height, the planners solved the problem 100 times, and their obtained results are shown in Fig. 6. It can be seen that the four planners needed approximately the same time to solve the problem when the gap percentage is equal to or bigger than 0.25%. However, they needed a different amount of time to find solutions for smaller gap height. Informed RRT*-Connect took the least time to return solutions among all planners. Informed RRT* was the second-fastest until the gap percentage was bigger than 0.001325% in which RRT*-Connect acted faster than Informed RRT* to return near-optimal solutions.

C. OMPL APP SIMULATIONS

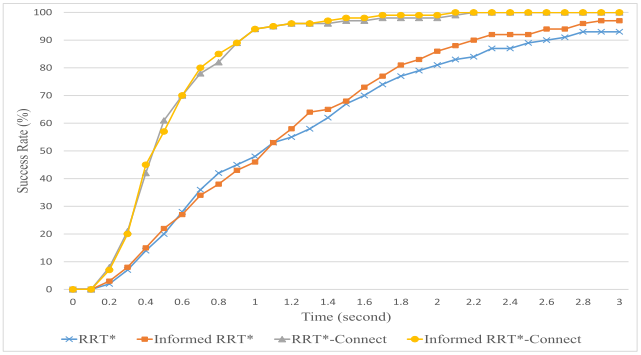
OMPL App is a front-end for OMPL, contains several configuration spaces and rigid body robots. We have used some of its configuration spaces, including 3D and 6D problems. The selected configuration spaces that have been tested in this paper are shown in Fig. 7.

1) MAZE PLANAR

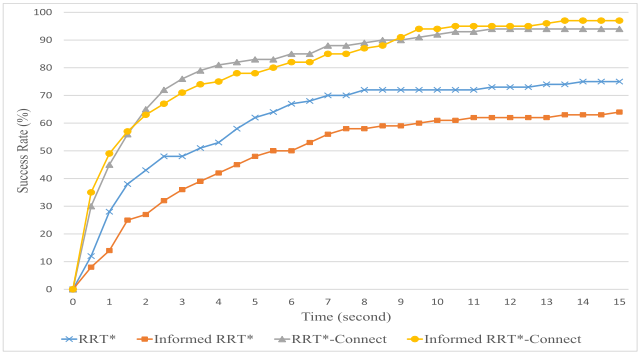
Maze Planar, Fig. 7a, is one of the OMPL App configuration spaces that has been designed for problems with 3DoFs, two real vectors (x-axis and y-axis) and rotation. The rigid body on the left side is the start pose, the red-colored shape, while the rigid body on the right side is the goal pose.

2) HOME

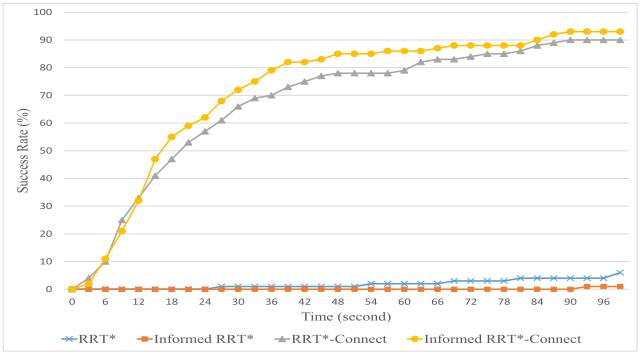
Home is a configuration space that has been designed for problems with 6DoFs (3 coordinate planes (x, y, z) and their rotations (roll, pitch, yaw)). Fig. 7b shows this configuration



(a) Maze Planar

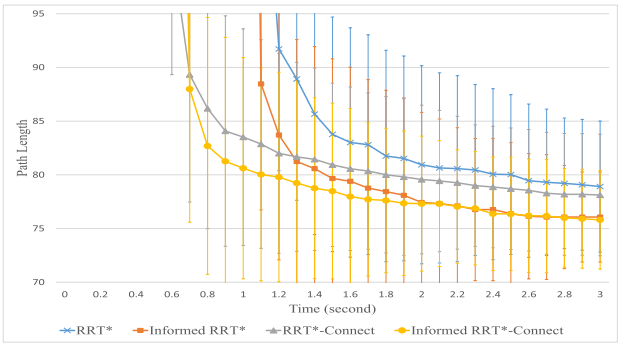


(b) Home

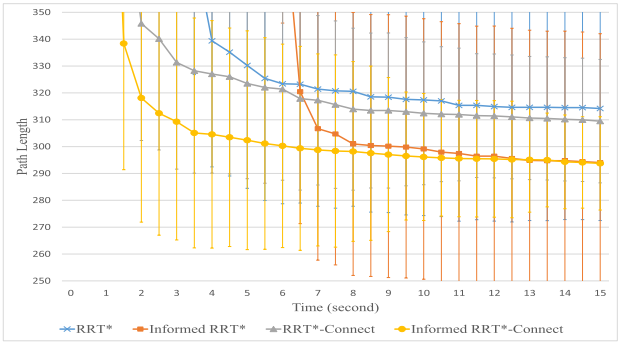


(c) Apartment Hard

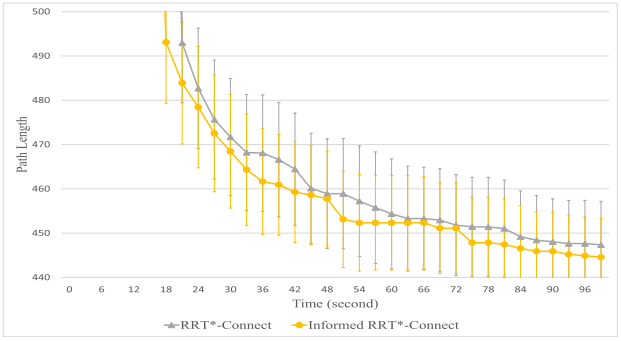
FIGURE 8. The success rate versus time of the four planners on OMPL App configuration spaces.



(a) Maze Planar



(b) Home



(c) Apartment Hard

FIGURE 9. The path length versus time of planners for different OMPL App configuration spaces. Error bars denote a nonparametric 95% confidence interval for the median path length.

space in which the table on the right side is the start pose, while the table on the left side shows the goal pose. In order to find near-optimal solutions, the planners have to pass through the windows located between the start pose and the goal pose.

3) APARTMENT HARD

The last simulation is named "*Apartment Hard*" by OMPL App developers, shown in Fig. 7c, in which start pose is the piano located on the left side of the map in the hall, while goal location is the piano which hidden beyond the kitchen walls on the right side of the map. To find near-optimal solutions, the planners need to pass the piano through the kitchen door.

V. EVALUATION

We present the evaluation of simulation in this section. First simulation, Single Cube scenario's goal is to show how the informed set acts on different map sizes. Fig. 4 shows that when the distance between x_{start} and x_{goal} is equal to the configuration space length, l, both planner acts similarly. It is due to the fact that when the distance between x_{start} and x_{goal} is relatively big and the subset almost covers all

over the configuration space so that Informed RRT*-Connect search area is not limited to a small portion of the configuration space. Therefore, Informed RRT*-Connect is acting like RRT*-Connect in these kinds of scenarios. In contrast, when the distance of x_{start} and x_{goal} is getting smaller, the time taken for the Informed version is significantly smaller (up to 10 times) than the standard version of RRT*-Connect.

The second simulation, which is about finding paths in a configuration space that only offers narrow passages to connect x_{start} and x_{goal} shows that dual-tree searches work better than the single-tree methods. It is shown that Informed RRT*-Connect acts better than Informed RRT*. Similarly, RRT*-Connect found paths faster than RRT*. Moreover, when the gap size was only 0.001325% of the map length, the tightest gap, RRT*-Connect worked faster than not only RRT* but also Informed RRT*. Informed RRT*-Connect worked as the fastest motion planner in all the gap sizes.

OMPL Apps configuration spaces' results are shown in Fig. 8 and Fig. 9. The success rate graphs, Fig. 8, show that the bidirectional searches, RRT*-Connect and Informed RRT*-Connect, are more successful than the unidirectional searches, RRT* and Informed RRT*. It is the significance of bidirectional planners over the unidirectional planners that are able to find solutions faster than unidirectional planners. It is also noticeable that Informed version of each planner work similar to the standard version in term of success rate. It is due to the fact that the informed version and the standard version act similarly before the first solution is found.

Informed RRT*-Connect and RRT*-Connect were able to achieve above 90% success in all three simulations, while Informed RRT* and RRT* were only able to achieve approximately 85%, 70%, and 5% in Maze Planner, Home, and Apartment Hard scenarios, respectively. Therefore, bidirectional searches are preferable for motion planning problems in which success in a limited time is essential.

Fig. 9 demonstrates the path length vs time in OMPL App configuration spaces achieved by the planners. In Fig. 9a and Fig. 9b, it can be seen that Informed versions of the planners produced better results in comparison with the standard versions. RRT* and Informed RRT* were not able to find solutions in Apartment Hard scenario so that for the path length graph, Fig. 9c, RRT*-Connect and Informed RRT*-Connect were shown. Informed RRT*-Connect was the most successful planner in terms of success rate and path length.

VI. CONCLUSION

This paper presented a new motion planner that combines the ability to quickly finding the first solutions from RRT*-Connect with the capability of quickly returning near-optimal solutions from Informed RRT*. Although RRT*-Connect is a fast path planner in terms of finding solutions, it scans all over the configuration space to return better solutions than the existing one like RRT*, which is not efficient, especially in high-dimensional problems.

The proposed method, Informed RRT*-Connect, not only find its first solutions as fast as RRT*-Connect, which is faster than single-tree based methods but also it returns near-optimal solutions quicker than RRT*-Connect. It is achieved by limiting the configuration space into an ellipsoid subset, which depends on the location of the start configuration, the goal configuration, and the length of the shortest path.

We have successfully demonstrated that the proposed method found first solutions similar to RRT*-Connect in the simulations, and it returns near-optimal solutions faster than the existing planners.

These properties make the proposed method suitable for the motion planning problems in which optimal solutions must be obtained with a limited number of iterations and/or in a limited time slot.

REFERENCES

[1] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.

[2] J.-C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *Int. J. Robot. Res.*, vol. 18, no. 11, pp. 1119–1128, Nov. 1999.

[3] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[4] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[5] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[6] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian processes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 1056–1062.

[7] X. Lan and S. Di Cairano, "Continuous curvature path planning for semi-autonomous vehicle maneuvers using RRT," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2015, pp. 2360–2365.

[8] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[9] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[10] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2997–3004.

[11] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 233–255, Mar. 2002.

[12] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

[13] S. M. Lavalle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[14] J. Kuffner and S. Lavalle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symposia*, vol. 2, Nov. 2002, pp. 995–1001.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[16] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "RRT∗-Connect: Faster, asymptotically optimal motion planning," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2015, pp. 1670–1677.

[17] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 966–984, Aug. 2018.

[18] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT∗," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2011, pp. 1478–1483.
[19] H. Sun and M. Farooq, "Note on the generation of random points uniformly distributed in hyper-ellipsoids," in *Proc. 5th Int. Conf. Inf. Fusion*, vol. 1, Jul. 2003, pp. 489–496.
[20] J. D. Gammell and T. D. Barfoot, "The probability density function of a transformation-based hyperellipsoid sampling technique," 2014, *arXiv:1404.1347*. [Online]. Available: https://arxiv.org/abs/1404.1347
[21] G. Wahba, "A least squares estimate of satellite attitude," *SIAM Rev.*, vol. 7, no. 3, p. 409, Jul. 1965.
[22] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.

REZA MASHAYEKHI received the bachelor's degree in electronics and the master's degree in electrical engineering (mechatronics and automatic control). He is currently pursuing the Ph.D. degree in computer science with the University of Malaya. His research interests are robotics, motion planning, machine learning, automatic control, and embedded systems.

MOHD YAMANI IDNA IDRIS (Member, IEEE) received the Ph.D. degree in electrical engineering. He has a vast experience in research. He is currently an Associate Professor with the Faculty of Computer Science and Information Technology, University of Malaya. His research interests are in the areas of robotics, embedded systems, sensor networks, and signal/image processing.

MOHAMMAD HOSSEIN ANISI (Senior Member, IEEE) worked as a Senior Research Associate with the University of East Anglia, U.K., and a Senior Lecturer with the University of Malaya, Malaysia. As a Computer Scientist, he has designed and developed novel architectures and routing protocols for the Internet of Things (IoT) enabling technologies, including wireless sensor and actuator networks, vehicular networks, heterogeneous networks, and body area networks. He has a strong collaboration with industry and working with several companies in U.K., with the focus on monitoring and automation systems based on the IoT concept capable of reliable and seamless generation, transmission, processing, and demonstration of data. He is currently an Assistant Professor with the School of Computer Science and Electronic Engineering, University of Essex, and the Head of the Internet of Everything Laboratory. He has published more than 80 articles in high-quality journals and several conference papers. His research has focused specifically on real world application domains, such as energy management, transportation, healthcare, and other potential life domains. His research results have directly contributed to the technology industry.

Dr. Anisi is a Fellow of Higher Education Academy. He has also been serving as an Executive/Technical Committee Member of several conferences. He is also a Technical Committee Member of Finnish-Russian University Cooperation in Telecommunications (FRUCT), a Senior Member of the Institute of Research Engineers and Doctors (the IRED), and a member of ACM, the IEEE Council on RFID, the IEEE Sensors Council, and the IEEE Systems Council and International Association of Engineers (IAENG). He received the Excellent Service Award for his achievements from the University of Malaya. He was a recipient of two medals for his innovations from PECIPTA 2015 and IIDEX 2016 expositions. He has received several International and national funding awards for his fundamental and practical research as PI and Co-I. He is also an Associate Editor of a number of journals, including the IEEE Access, *Ad Hoc & Sensor Wireless Networks*, *IET Wireless Sensor Systems*, the *International Journal of Distributed Sensor Networks*. the *KSII Transactions on Internet and Information Systems journals*, and the *Journal of Sensor and Actuator Networks*. He has been a Guest Editor of special issues of the journals and a Lead Organizer of special sessions and workshops at the IEEE conferences, such as the IEEE CAMAD, the IEEE PIMRC, and the IEEE VTC.

ISMAIL AHMEDY (Member, IEEE) received the Ph.D. degree in computer science. He is currently a Senior Lecturer with the Department of Computer Systems and Technology, Faculty Science Computer and Information Technology, University of Malaya. His areas of expertise are underwater acoustic sensor networks, embedded systems, and wireless sensor networks.

IHSAN ALI (Student Member, IEEE) received the M.S. degree in computer system engineering from the GIK Institute, in 2008. He is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology, University of Malaya.

He is also an Active Researcher (a Research Associate) with the Centre for Mobile Cloud Computing Research (C4MCCR), Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia. He has published several high-impact research journal articles, including a highly reputable *IEEE Communications Magazine*. He has been actively involved in research and teaching activities, for the last ten years in different countries, including Saudi Arabia, USA, Pakistan, and Malaysia. His research interests include wireless sensor networks, robotics in WSNs, sensor cloud, fog computing, the IoT, ML/DL in wireless sensor networks.

Dr. Ali has served as a Technical Program Committee Member for the IWCMC 2017-2018, AINIS 2017, Future 5V 2017, ICACCI-2018, and INAIT 2019, and also an Organizer of the Special session on fog computing in Future 5V 2017. He is also an Active Reviewer of computers and electrical engineering, the *KSII Transactions on Internet and Information Systems*, *Mobile Networks and Applications*, the *International Journal of Distributed Sensor Networks*, the *Journal of Advanced Transportation*, the IEEE Transactions on Intelligent Transportation Systems, *Computer Networks*, IEEE Access, *Wireless Communications and Mobile Computing*, and the *IEEE Communications Magazine*.