IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 22, 2019, accepted September 11, 2019, date of publication September 16, 2019, date of current version October 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2941537

A Multi-Heuristic A* Algorithm Based on Stagnation Detection for Path Planning of Manipulators in Cluttered Environments

KAI MI[1,2], JUN ZHENG[1], YUNKUAN WANG[1], AND JIANHUA HU[1]

[1]Intelligent Manufacturing Technology and System Research Center, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[2]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Jun Zheng (jun.zheng@ia.ac.cn)

This work was supported by the Major Science and Technology Project of Henan Province under Grant 161100210300.

ABSTRACT We consider the problem of planning an obstacle avoidance path for manipulators in cluttered environments especially with narrow passages. Compared to sampling-based planners, heuristic search-based planners are more suitable for such environments due to the consistent heuristic guidance. In order to solve the problem of search stagnation caused by inappropriate heuristic guidance, we use the Shared Multi-Heuristic A* (SMHA*) algorithm and predefine multiple inadmissible heuristics. Meanwhile, when the consistent heuristic guidance is correct and appropriate, in order to avoid the unnecessary inadmissible heuristics to increase the search burden, we improve it by adding heuristic-based stagnation detection for each extended node and the improved algorithm is called SD-SMHA*. Only when the algorithm detects that it ceases to make significant progress towards the goal, the predefined inadmissible heuristics will be introduced. Finally, multiple simulation experiments are carried out and the results show that the improved algorithm effectively improves the planning efficiency and planning success rate in different scenarios.

INDEX TERMS Path planning, obstacle avoidance, search-based planner, stagnation detection, multi-heuristic A*.

I. INTRODUCTION

In recent years, with the increasing use of manipulators, their application scenarios have become increasingly complex. Especially for some special tasks, the manipulator needs to pick or place objects through narrow passages. Given an initial and goal configurations of the manipulator, the objective of path planning is to plan a continuous and feasible path while avoiding obstacles along the way. Many path planning algorithms have been proposed for different application scenarios and requirements. Planning time, planning success rates and the quality of planning trajectories have become important indicators for measuring these planning algorithms.

Due to the completeness of probability, sampling-based motion planning algorithms are increasingly favored by scholars, especially in high-dimensional applications. Typical examples include Probabilistic Road Maps (PRM) [1] and Rapidly-exploring Random Trees (RRTs) [2], meanwhile a large number of improved algorithms have also been proposed based on them. Kuffner and LaValle [3] used two bidirectional random trees named RRT-Connect to speed up the planning process. In order to improve the path cost, Urmson and Simmons proposed a Heuristically Biasing RRT (hRRT) [4] algorithm which added cost value for each expansion node and biased to connecting nodes with lower costs. However, h-RRTs could not guarantee optimal solution. In order to obtain the asymptotic optimality, Karaman and Frazzoli [5] proposed RRT* algorithm, which introduced an optimization structure based on RRT algorithm. Due to the randomness of exploration, paths planned by such algorithms have poor consistency. Given the same initial state and goal state, the path planned by the same algorithm is greatly different each time. In addition, RRT* algorithm continuously optimizes the connection during the node expansion process. For complex application environments, especially in narrow passages, it takes longer to obtain optimized solutions and has lower planning efficiency.

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

In order to plan an optimized trajectory faster, many algorithms based on trajectory optimization are proposed and they are essentially evolved on the basis of the potential field method [6]–[8]. These early potential-field based methods are sensitive to local extremes and not suitable for high dimensional applications such as manipulators or humanoid robots. Recently Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [9] revives interest in trajectory optimization methods and proposes a novel formulation of trajectory costs. CHOMP takes trajectory smoothing and obstacle avoidance as the optimization objectives and describes them into the trajectory cost mathematically. Many improved algorithms [10]–[12] based on it have been proposed. One of the main drawbacks of them is that more dense trajectory states need to be described in order to reason about obstacles in complex environments and it will increase the computational cost significantly. To overcome this problem, the Gaussian process motion planning family of algorithms [13]–[15] samples a few states on the initial trajectory and use Gaussian interpolation to query the trajectory at any time of interest. It effectively speeds up the trajectory optimization. However, such algorithms are not complete, and their planning results depend on the given initial trajectory. For complex scenarios, especially with narrow channels, the algorithms are easy to fall into local extremes and the planning success rate is greatly reduced.

The graph search-based planning algorithms represented by A* [16] are widely used in two-dimensional (2D) or three-dimensional (3D) space. For high dimensional applications such as manipulators or humanoid robots, the search complexity will rise dramatically. To this end, Cohen *et al.* [17], [18] improved the original algorithms from three perspectives: heuristic mode, motion primitives and search function, which effectively speeded up the search speed in high-dimensional space. At the same time, the proposed algorithm has completeness and bounds on sub-optimality. The algorithm establishes heuristic guidance in the 3D workspace, which facilitates the manipulators to pass through a narrow and complex environment. However, due to the lack of structural information of the manipulators in the establishment of heuristic guidance, the heuristic guidance is sometimes incorrect and it will lead to search stagnation.

In this paper, the heuristic search-based algorithm will be used to solve the problem of path planning for manipulators in cluttered environments, especially with narrow passages. In order to solve the problem of heuristic guidance error, we use the Shared Multi-Heuristic A* (SMHA*) [19] algorithm for path searching where one consistent heuristic coexists with multiple inadmissible heuristics. Different from the original SMHA* algorithm, we introduce stagnation detection for each extended node during the search process and an improved algorithm is proposed which we called SD-SMHA*. Only when the consistent search is detected to be stagnant, the inadmissible heuristics will be introduced. Meanwhile, when one of the inadmissible searches is stagnant, the corresponding inadmissible heuristic will be cancelled. By introducing a stagnation detection on the basis of SMHA*, the SD-SMHA* algorithm will avoid unnecessary inadmissible searches and improves the search efficiency effectively.

The remainder of this article is organized as follows. The second section discusses the related work about the motion planning problems in cluttered environments. Some Preliminary materials including problem definition and a brief description of the original SMHA* algorithm are presented in Section III. Section IV presents and describes the SD-SMHA* algorithm in detail from two aspects: stagnation detection and the specific process of the algorithm. Then a detailed theoretical analysis including algorithm completeness, bounded sub-optimality and bounded re-expansions is presented in Section V. Section VI illustrates some implementation details of the proposed algorithm in the application of manipulators. Then, we present simulation experiments for the 6 degree of freedom (DOF) manipulator in two scenes with narrow passages and the experimental results confirm the effectiveness and superiority of the proposed algorithm in Section VII. Finally, Section VIII concludes the paper and also suggests some improvement directions of the current algorithm.

II. RELATED WORK

For cluttered environments, especially those with narrow passages, the algorithms based on trajectory optimization are easy to fall into local extremes due to their incompleteness, and the planning success rate is low. Huang *et al.* [20] use multiple initial trajectories to form a graph, and trajectory optimization is directly performed on the graph, thereby planning multiple effective trajectories and improving the success rate obviously. However, the algorithm is only applied for mobile robots in 2D space. It is difficult to give multiple suitable initial trajectories in high-dimensional space and not applicable for manipulators.

Many improved sampling-based algorithms [21], [22] are proposed for motion planning in cluttered environments due to their completeness. However these algorithms are only suitable for mobile robots in 2D space and the planning path is not optimal. In order to plan an optimal path, various algorithms [23]–[25] enhancing original RRT* planner [5] have been proposed. Qureshi AH et al. introduced intelligent sample insertion heuristic based on Bidirectional-RRT* algorithm [26] and proposed an improved planner called Intelligent Bidirectional-RRT* (IB-RRT*) [24]. Later, Zaid *et al.* [25] added potential field guidance on this basis and a new planner was proposed called Potentially Guided Intelligent Bidirectional-RRT* (PIB-RRT*). However, all of these algorithms are only simulated in 2D or 3D spaces and are difficult to implement in high-dimensional space. For example, it is challenging to establish potential field guidance in the configuration space of manipulators.

The improved heuristic search-based algorithm proposed by Cohen *et al.* [17] uses a 3D breadth-first search (BFS) in

the workspace to establish heuristic guidance for the effector of manipulator. The heuristic search guidance is conducive to manipulators through cluttered environment. However due to the lack of structural information of the manipulator, the heuristic constructed by BFS sometimes leads the search to an area that the manipulator cannot reach. For this problem, Ariyan *et al.* [27] developed a context dependent search strategy switching (CODES3) algorithm which estimates the degree of congestion for each extended node in configuration space by sampling. Although the estimation of congestion is performed in parallel to the search algorithm, the task is still heavy and takes a lot of time. The Multi-Heuristic A* algorithm [19] developed by Sandip Aine et al is an effective and popular approach. By adding multiple inadmissible heuristics, the algorithm can effectively avoid search stagnation caused by incomplete or erroneous consistent heuristic. However, when the consistent heuristic is right and complete, the extra inadmissible heuristics will reduce search efficiency. Meanwhile, the incorrect inadmissible heuristics can also cause the algorithm to stagnate and greatly increase search time. For this problem, Islam *et al.* [28] proposed a method to seek user guidance when the planner identifies that it ceases to make significant progress towards the goal. This method is suitable for more complex humanoid robots. For manipulator applications, it increases planning difficulty and is less intelligent.

This paper proposes an improved shared multi-heuristic A* search algorithm based on stagnation detection which we called SD-SMHA* algorithm. The specific contributions of this paper are as follows.

1) A heuristic-based search stagnation detection method is adopted and the effectiveness of the detection is verified;
2) The stagnation detection is introduced into the original SMHA* algorithm and is performed for each expansion node including the inadmissible search. It greatly improves the search efficiency and planning success rate;
3) Detailed theoretical analysis for the proposed SD-SMHA* algorithm including its completeness, bounded sub-optimality and bounded re-expansions is presented.

III. PRELIMINARY MATERIALS

In this section, we will formulate the problem and give some necessary definitions and terms. Then a brief description of SMHA* algorithm will be presented.

A. PROBLEM DEFINITION

For a manipulator with n degrees of freedom (DOF), we define C to be its n-dimensional configuration space. An n-dimensional configuration q is free, if the manipulator placed at q has no collision with obstacles in the environment and with itself. Then we define C_{free} as free space which is a set of all the free configurations in C. Corresponding $C_{obs} = C/C_{free}$ indicates obstacle space. Given a initial configuration $q_{start} \in C_{free}$ and a goal configuration $q_{goal} \in C_{free}$, the path planning problem is to find a continuous path in C_{free} connecting q_{start} with q_{goal}. In this article, we put forward higher requirements for this problem as follows.

1) The motion planning is performed in high-dimensional configuration space and the planning scenario is cluttered, especially with narrow passages;
2) Completeness. If there is a feasible path in C_{free}, it can be found given enough time;
3) Boundary sub-optimality. For a planning problem, we define $g^*(q_{goal})$ as the cost of the optimal path, and set the boundary parameter to ω, then the problem is to find a sub-optimal path, so that $g(q_{goal}) \leq \omega * g^*(q_{goal})$.

Algorithm 1 Shared Multi-Heuristic A*(SMHA*)

```
1:  g(s_start) = 0;  g(s_goal) = ∞
2:  bp(s_start) = bp(s_goal) = null
3:  v(s_start) = v(s_goal) = ∞
4:  for i = 0, 1, ··· , n do
5:     OPEN_i ← Φ
6:     Insert s_start in OPEN_i with Key(s_start , i)
7:  CLOSED_anchor ← Φ
8:  CLOSED_inad ← Φ
9:  while OPEN_0.Minkey() <∞ do
10:    for i = 1, 2, ··· , n do
11:       if OPEN_i.Minkey() ≤ ω_2 * OPEN_0.Minkey() then
12:          if g(s_goal) ≤ OPEN_i.Minkey() then
13:             if g(s_goal) ≤ ∞ then
14:                Terminate and return path pointed by
   bp(s_goal)
15:          else
16:             s ← OPEN_i.Top()
17:             ExpandState(s)
18:             Insert s in CLOSED_inad
19:       else
20:          if g(s_goal) ≤ OPEN_0.Minkey() then
21:             if g(s_goal) ≤ ∞ then
22:                Terminate and return path pointed by
   bp(s_goal)
23:          else
24:             s ← OPEN_0.Top()
25:             ExpandState(s)
26:             Insert s in CLOSED_anchor
```

B. SHARED MULTI-HEURISTIC A* (SMHA*)

Shared Multi-heuristic A* algorithm is a search-based planning algorithm that takes in multiple inadmissible heuristic functions in addition to a single consistent heuristic termed the anchor heuristic. Algorithm 1 outlines the pseudo-code of SMHA*. The procedures used in Algorithm 1 are described below.

Key(s, i): This function returns estimation of the total cost from start to goal going through s corresponding to the *ith* heuristic function. The specific calculation is as follows:

$$Key(s, i) = g(s) + \omega_1 * h_i(s), \tag{1}$$

Algorithm 2 Stagnation Detection(Q_i, s)

1: **for** $m = 1, 2, \cdots, \sigma_1 - 1$ **do**
2: $\quad Q_i(m) = Q_i(m+1)$
3: **end for**
4: $Q_i(\sigma_1) = s$
5: $\eta_{Q_i}(1, \sigma_2) = min_{1 \le n \le \sigma_2}\{h_i(Q_i(n))\}$
6: $\eta_{Q_i}(\sigma_2, \sigma_1) = min_{\sigma_2 \le n \le \sigma_1}\{h_i(Q_i(n))\}$
7: **if** $\eta_{Q_i}(\sigma_2, \sigma_1) \ge \eta_{Q_i}(1, \sigma_2) - \varepsilon$ **then**
8: **return** Ture
9: **else**
10: **return** False
11: **end if**

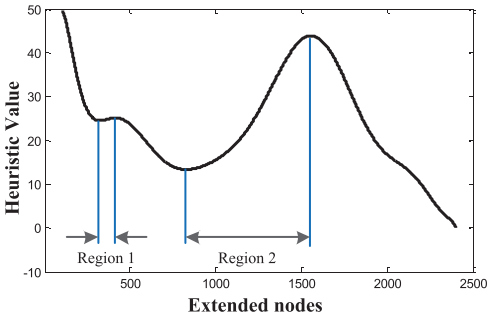


FIGURE 1. The curve of heuristic cost for extended nodes during search. The region 1 and region 2 are two *heuristic depression* regions.

where $g(s)$ represents the cost of the current path from the start node to s, ω_1 is a inflation factor as shown in Weighted A* (WA*)[29], and $h_i(s)$ denotes the *ith* heuristic function and $h_0(s)$ is the single consistent heuristic function.

ExpandState(s): This procedure is done in a similar way to WA*. The main difference is that each heuristic function corresponds to an open list ($OPEN_i$), and the extension nodes are shared into each open list as long as some conditions are met. So if a better path to a state s is discovered by any of the heuristic searches, the new cost-to-come values ($g(s)$) will be updated in all of the open lists.

The SMHA* algorithm presented in Algorithm 1 is also complete like its baseline algorithm WA*. By introducing multiple inadmissible heuristics, SMHA* performs node expansion under different guidance and can break away from search stagnation areas effectively. As shown in line 11 of the pseudo-code, supposing that s_a and s_b is the nodes with the lowest cost in $OPEN_i$ and $OPEN_0$ respectively, only when $Key(s_a, i) < \omega_2 * Key(s_b, 0)$, the nodes in $OPEN_i$ will be expanded. This ensures that the algorithm can only terminate with a solution within $\omega_1 * \omega_2$ bound ($\omega_2 \ge 1$), otherwise no finite cost solution exists. The detailed proof of the boundary sub-optimality for the algorithm can be found in [19].

IV. ALGORITHM DESCRIPTION

In this section, we will explain the heuristic-based search stagnation detection method and analyze its effectiveness firstly. Then, an improved shared multi-heuristic A* search algorithm based on stagnation detection which we called SD-SMHA* will be elaborated in detail.

A. HEURISTIC-BASED STAGNATION DETECTION

For some specific scenarios, the path search inevitably falls into a stagnation region where the planner ceases to make progress. For 2D or 3D applications, the search space is relatively small and it can escape this region after a certain period of time. However for high-dimensional applications, the time required is unacceptable. So it is necessary to detect the stagnation regions. Search vacillation and heuristic depressions are the two main features when the planner ceases to make progress. For the former, a vacillation-based detection method [30] can be employed which uses the notion of expansion delays. However in actual experiments, we found that the noise of the extended delay is relatively large, and sometimes false detection occurs. So we adopt the heuristic-based stagnation detection method corresponding to the later feature. The pseudo-code of Stagnation Detection is shown in Algorithm 2.

Let σ_1, σ_2 be parameters and $\sigma_1 > \sigma_2$, Q_i be a queue scrolling storage the past σ_1 extended nodes corresponding to the heuristic function $h_i()$. We define $\eta_{Q_i}(1, \sigma) = min_{1 \le n \le \sigma}\{h_i(Q_i(n))\}$ and set a threshold $\varepsilon(\varepsilon \ge 0)$. The pseudo-code of Stagnation Detection is shown in Algorithm 2. We only care about the past σ_1 extended nodes and roll updates the nodes in the queue Q_i in Line1-4. We believe that the heuristic search $h_i()$ is stuck in a stagnant region, if the minimum heuristic cost ($\eta_{Q_i}(\sigma_2, \sigma_1)$) of the last $\sigma_1 - \sigma_2$ extended nodes is not reduced or decreased less than the threshold ε, compared to the previous σ_2 extended nodes ($\eta_{Q_i}(1, \sigma_2)$) (Line 5-11).

In order to verify the effectiveness of the detection method, we give an example for illustration as shown in Figure 1 and Figure 2. Figure 1 shows the change of the heuristic cost for the extended nodes during searching. Region 1 and region 2 labeled in the figure are two heuristically depressed regions and it can be detected by our method as shown in Figure 2(a). The region 1 is relatively small and it can be filtered out by adjusting the values of σ_1 and σ_2. As shown in Figure 2(b), the difference between σ_1 and σ_2 is increased from 150 to 200 and the small stagnation region (Region 1 in Fig 1) has been filtered out. The disadvantage is that the stagnation detection is relatively delayed.

B. SD-SMHA* ALGORITHM

Here, we introduce the heuristic-based stagnation detection into the Shared Multi-Heuristic A* and propose an improved algorithm which we called SD-SMHA*. The detailed pseudo-code is shown in Algorithm 3.

The algorithm begins with various initializations in Line 2-12. The variable $detecte\_h_0$ represents whether the consistent search is detected for stagnation and it is initialized with True. The variable $EnableExpand(i)$ indicates whether

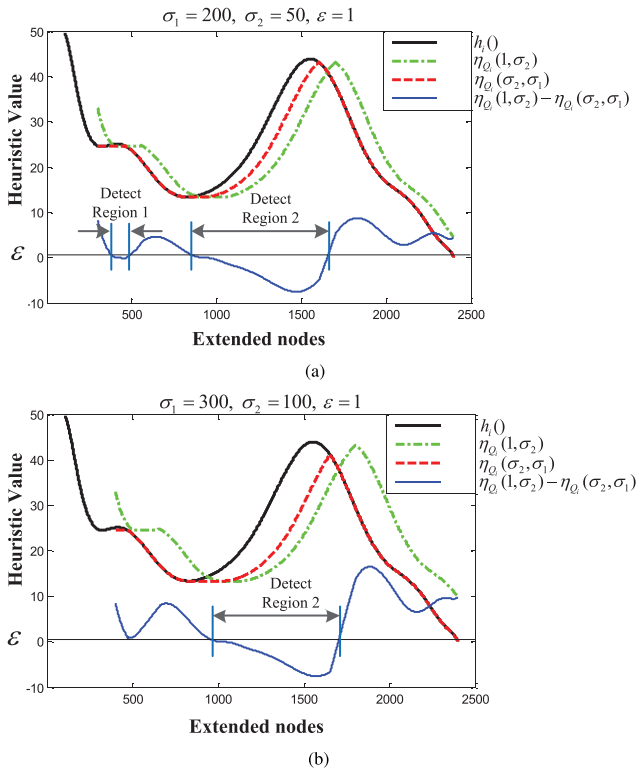


FIGURE 2. The results of heuristic-based stagnation detection with different parameters. The difference between σ_1 and σ_2 is 150 in image (a) and it has been enlarged to 200 in image (b).

the heuristic function $h_i()$ allows extension and it is initialized to false for all inadmissible heuristics. So at the beginning, the algorithm only performs a search expansion under the consistent heuristic function $h_0()$ as shown in Line 15. Each expansion will perform stagnation detection on the current extended node (Line 33). If the node is detected not falling into a stagnant region, then the algorithm runs the same as Weighted A*, and the only difference is that in the procedure of ExpandState(s) in algorithm 4, the extended nodes will be shared into the open list of inadmissible heuristics that satisfy the condition in Line 12 of algorithm 4. If the node is detected falling into a stagnant region, all of variable $EnableExpand(i)$ will be set to True in Line 34-35 and the variable $detecte\_h_0$ is set to False in Line 37. Then all inadmissible heuristics will be enable and the stagnation detection for the consistent heuristic will no longer occur which is reflected by the variable $detecte\_h_0$ (Line 33-37). Later, the algorithm will perform stagnation detection on each node which is expanded by inadmissible searches (Line 23). If the search is detected falling into a stagnant region, the corresponding variable $EnableExpand(i)$ will be set to False again and the corresponding inadmissible heuristic will be cancelled (Line 15 and 23). Whether for the consistent heuristic or inadmissible heuristics, as long as the cost value $g(s_{goal})$ of the goal is less than the minimum value of the nodes in the current open list, the feasible solution is searched and the algorithm terminates (Line 16-18 and Line 26-28).

Algorithm 3 SD-SMHA*

1: **procedure** Main()
2: $g(s_{start}) = 0;\ \ g(s_{goal}) = \infty$
3: $bp(s_{start}) = bp(s_{goal}) = null$
4: $detect\_h_0$ = True
5: **for** $i = 0, 1, \cdots, n$ **do**
6: $EnableExpand(i)$= False
7: $Q_i \leftarrow \Phi$
8: $OPEN_i \leftarrow \Phi$
9: Insert s_{start} in $OPEN_i$ with Key(s_{start}, i)
10: **end for**
11: $CLOSED_{anchor} \leftarrow \Phi$
12: $CLOSED_{inad} \leftarrow \Phi$
13: **while** $OPEN_0.Minkey() < \infty$ **do**
14: **for** $i = 1, 2, \cdots, n$ **do**
15: **if** $OPEN_i.Minkey() \leq \omega_2 * OPEN_0.Minkey()$ **and** $EnableExpand(i)$ **then**
16: **if** $g(s_{goal}) \leq OPEN_i.Minkey()$ **then**
17: **if** $g(s_{goal}) \leq \infty$ **then**
18: Return path pointed by $bp(s_{goal})$
19: **else**
20: $s \leftarrow OPEN_i.Top()$
21: ExpandState(s)
22: Insert s in $CLOSED_{inad}$
23: $EnableExpand(i)$=!StagnationDectection(Q_i, s)
24: **end if**
25: **else**
26: **if** $g(s_{goal}) \leq OPEN_0.Minkey()$ **then**
27: **if** $g(s_{goal}) \leq \infty$ **then**
28: Return path pointed by $bp(s_{goal})$
29: **else**
30: $s \leftarrow OPEN_0.Top()$
31: ExpandState(s)
32: Insert s in $CLOSED_{anchor}$
33: **if** $detect\_h_0$ **and** StagnationDectection(Q_0, s)
34: **for** $i = 1, 2, \cdots, n$ **do**
35: $EnableExpand(i)$= True
36: **end for**
37: $detect\_h_0$= False
38: **end if**
39: **end if**
40: **end for**
41: **end while**
42:
43: **procedure** Key(s,i)
44: **return** $g(s) + \omega_1 * h_i(s)$

As mentioned above, we performed stagnation detection on the nodes guided by the consistent heuristic and the inadmissible heuristics. Firstly, for some scenarios, the consistent heuristic is right and complete, the algorithm will run the same as Weighted A* and it will greatly reduce the extra search guided by inadmissible heuristics in the original SMHA* algorithm. So the planning time will decrease obviously. In addition, when the consistent heuristic is inappropri-

Algorithm 4 ExpandState(s)

```
1: Remove s from OPEN_i ∀i = 0, 1, ··· , n
2: for all s' ∈ Succ(s) do
3:    if s' was never gengerated then
4:       g(s') = ∞; bp(s') = null
5:    end if
6:    if g(s') > g(s) + c(s, s') then
7:       g(s') = g(s) + c(s, s'); bp(s') = s
8:       if s' ∉ CLOSED_anchor then
9:          Insert/Update s' in OPEN_0 with Key(s',0)
10:         if s' ∉ CLOSED_inad then
11:            for i = 1, 2, ··· , n do
12:               if Key(s', i) ≤ ω2 * Key(s', 0) then
13:                  Insert/Update s' in OPEN_i with Key(s', i)
14:            end for
15:   end if
16: end for
```

ate, the search will fall into stagnation in WA*. Here we add stagnation detection on the consistent search and introduce multiple inadmissible heuristics. The planning success rate will be greatly improved. Meanwhile, if the inadmissible heuristic is wrong, the search guided by the inadmissible heuristic will stagnate and it is meaningless. In our planner, the meaningless inadmissible search will be cancelled due to the stagnation detection and it will also increase the algorithm's fault tolerance for the inadmissible heuristics.

V. THEORETICAL ANALYSIS

The SD-SMHA* have guarantees similar to SMHA*. Firstly, the algorithm is complete. Then the sub-optimality of the solution is bounded by $\omega_1 * \omega_2$ times the cost of the optimal solution and no state is expanded more than twice (at most once by the consistent search and once by any inadmissible searches).

Theorem 1 (Completeness): SD-SMHA* is complete with respect to the search graph.

Proof: Depending on whether the consistent search is stuck in a stagnant region, the SD-SMHA* algorithm will run similar to Weighted A* and SMHA* respectively. Both Weighted A* and SMHA* algorithm have been proven to be complete with respect to the search graph. So the algorithm we proposed is also complete.

Theorem 2 (Bounded Sub-Optimality): If a feasible solution is found by SD-SMHA*, $g(s_{goal}) \leq \omega_1 * \omega_2 * g^*(s_{goal})$. In the other word, the cost of the solution is at most $\omega_1 * \omega_2$ times the cost of the optimal.

Proof: If the consistent search is detected not falling into a stagnant region throughout the search process, the algorithm will run similar to Weighted A* and the cost of solution must be less than ω_1 times the cost of the optimal solution ($g(s_{goal}) \leq \omega_1 * g^*(s_{goal})$). It has been fully demonstrated in [29]. On the contrary, if the consistent search is detected falling into a stagnant region, the inadmissible heuristics will be used and the algorithm will run similar to SMHA*. The only difference is the increased stagnation detection for each inadmissible search and it cannot affect the cost of the solution. Therefore, the proof of bounded sub-optimality for SMHA* in [19] can also be used here and we have $g(s_{goal}) \leq \omega_1 * \omega_2 * g^*(s_{goal})$. In the two cases mentioned above, $\omega_2 \geq 1$ and we have the solution cost to be within $\omega_1 * \omega_2$ factor of the optimal solution cost. The bounded sub-optimality of SD-SMHA* has been proved.

Theorem 3 (Bounded Re-Expansions): No state is expanded more than twice (at most once by the consistent search and once by one of the inadmissible searches).

Proof: Similarly, we explain it separately from the two situations. If the consistent search is detected not falling into a stagnant region, the inadmissible searches will not appear as shown in Line 14 and any states can only be expanded by the consistent search once (Line 30 and Line 44). On the contrary, if the consistent search is detected falling into a stagnant region, all of the inadmissible heuristic will be added. If a node has been expanded by the consistent search, it will be added into CLOSED_{anchor} at Line 30 and will no longer be expanded by any searches as shown in Line 38 and Line 44. In addition, if a node has been expanded by any inadmissible search, then it will be added into CLOSED_{inad} at Line 21 and will no longer be expanded by any inadmissible searches as shown in Line 38 and Line 46. Since the node has not been added into CLOSED_{anchor}, it can be expanded again only when the cost of the node ($g(s)$) expanded by the consistent search is less than the original as shown in Line 42 and Line 44. In summary, any node will only be expanded at most twice.

VI. IMPLEMENTATION DETAIL

In this section, we will elaborate on the implementation of the algorithm for manipulators from the perspectives of motion primitives, heuristics, and cost function.

A. MOTION PRIMITIVES

For robotic applications, in order to ensure the completeness, we search in the configuration space and adopt the motion primitives which have been designed in [17]. Each motion primitives represent a smallest possible motion that can be performed at any given state and it is the difference in the global joint angles of neighboring states. So for any state, the motion primitives are the same and we can pre-compute them offline. We call these smallest constant motions *static motion primitives*. In our experiments for a 6-DOF manipulator, 12 basic motion primitives are used along with six additional motion primitives. Each joint is oriented at a predefined angle in the forward or reverse direction and it constitutes 12 basic motion primitives. In order to speed up the search for the configuration space, the six additional motion primitives are designed for the first three joints with larger specified angle as shown in Figure 3. For many applications, the goal is to move the end effector to a specified pose. It is time consuming to search to the goal pose accurately using *static motion primitives*. So when a state s which is being

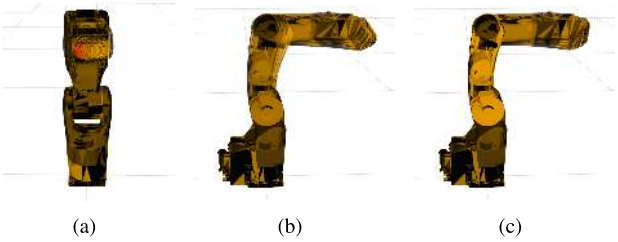


FIGURE 3. Three additional motion primitives for the first three joints which move in the forward direction are shown here. Each joint is rolled 7° and it is represented by (7°, 0, 0, 0, 0, 0) for image (a).

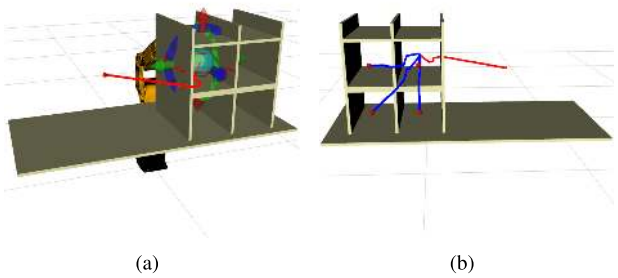


FIGURE 4. The heuristic guide curves for the start state including consistent heuristic (red) and inadmissible heuristics (blue). The red curve in image (a) is the consistent heuristic for the start state. In image (b), the red points are the auxiliary goal positions and the corresponding inadmissible heuristics are the blue curves.

expanded and its corresponding end-effector position is close to the goal, we generate a motion primitive connecting s to the goal state s_{goal} which is solved by an inverse kinematics (IK) solver directly. Here we call it *adaptive motion primitive*.

B. HEURISTICS

The algorithm proposed in this paper is essentially a heuristic search algorithm and the correctness of the heuristic function determines the search efficiency. We use the multi-heuristic approach which includes one consistent (anchor) heuristic h_{anch} and multiple inadmissible heuristics h_{inad}. The consistent heuristic is designed for the end effector and guides it to the specified goal position in the workspace. Before searching, we use breadth-first search (BFS) to calculate the shortest distance which represents the heuristic cost from any point in the workspace to the goal position. Then, the shortest obstacle avoidance path of the end effector to the goal position in any state of the manipulator is known and we take it as consistent heuristic to guide the searching as shown in Figure 4(a). Obviously the heuristic is not correct in this scenario and some auxiliary inadmissible heuristics are needed.

For the inadmissible heuristic, it includes guiding the search to a pre-define posture or position. In our implementation, we mainly set four auxiliary goal positions for the current task. As shown in Figure 4(b), the four red points before the shelf are the auxiliary goal positions and the blue lines are the corresponding inadmissible heuristics from start state to the auxiliary goals which is also obtained by BFS. We define one of the inadmissible heuristic costs as follows:

$$h_{inad}(s) = \begin{cases} h_{q_i}(s) + h_{goal}(q_i) & q_i \notin ANCESTOR(s) \\ h_{goal}(s) & q_i \in ANCESTOR(s) \end{cases} \quad (2)$$

where q_i is the *ith* auxiliary goal position, $h_{q_i}(s)$ is the estimated cost of the current extended node s to q_i, $h_{goal}(q_i)$ is the estimated cost from q_i to the goal, $ANCESTOR(s)$ is a collection of all the ancestors of node s. Then, $h_{inad}(s)$ represents the estimated cost from s to the goal via auxiliary goal position q_i as defined in [31].

C. COST FUNCTION

For heuristic search, the order of node expansion is determined by its cost value. As shown in Line 36 of Algorithm 3, the cost function is defined by

$$f_i(s) = g(s) + \omega_1 * h_i(s) \quad (3)$$

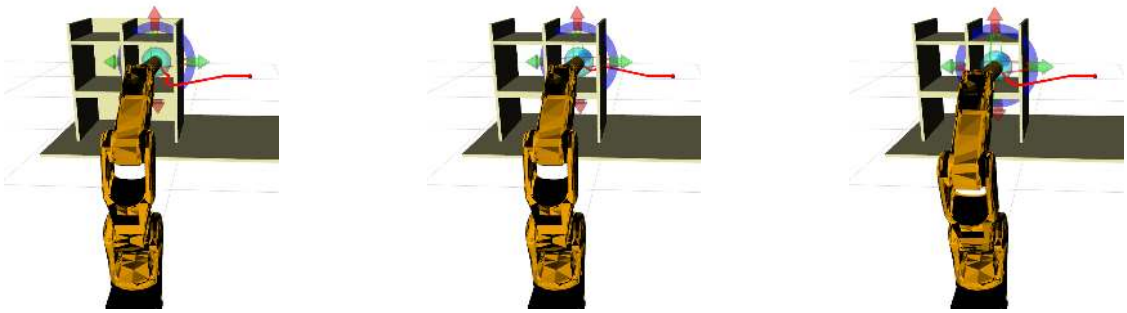
where $g(s) = g(s') + c(s', s)$. $g(s')$ represents the actual cost from the start node to the parent node and $c(s', s)$ is the expansion cost which is valued by the joint distance of the corresponding motion primitive in our implementation. $h_i(s)$ represents the heuristic cost for different heuristic functions and it is estimated by the length of the shortest obstacle avoidance path from the current position to the (auxiliary) goal position of the end effector. Each node has a different cost for different heuristics, and it determines the extension order for the node in different open lists $OPEN_i$. Finally, ω_1 is the same as defined in Weight A*[29]. Larger value of ω_1 will speed up the search to the goal and, in turn, increase the suboptimal boundary of the solution.

VII. EXPERIMENTAL RESULTS

In order to verify the validity and superiority of the algorithm, we evaluate the performance of the SD-SMHA* algorithm on a 6-DOF manipulator in different application scenarios. We use the Denso robot as a simulation model and add a cylindrical end-effector with a length of 0.3 meter for easy access to narrow passages. We use 12 basic motion primitives with a step size of 2° and six additional motion primitives with a step size of 4°. In addition, we set the threshold of the *adaptive motion primitive* to 0.01m. Only when the position of the end effector is searched to be less than 0.01m from the goal, the *adaptive motion primitive* can be used. In order to estimate the cost of the heuristics by BFS, the workspace is rasterized in advance with a grid accuracy of 0.01m. All experiments are performed on a personal computer with 3.2 GHz Intel i7-8700 CPU and 8 GB memory under Ubuntu 1604 operating system.

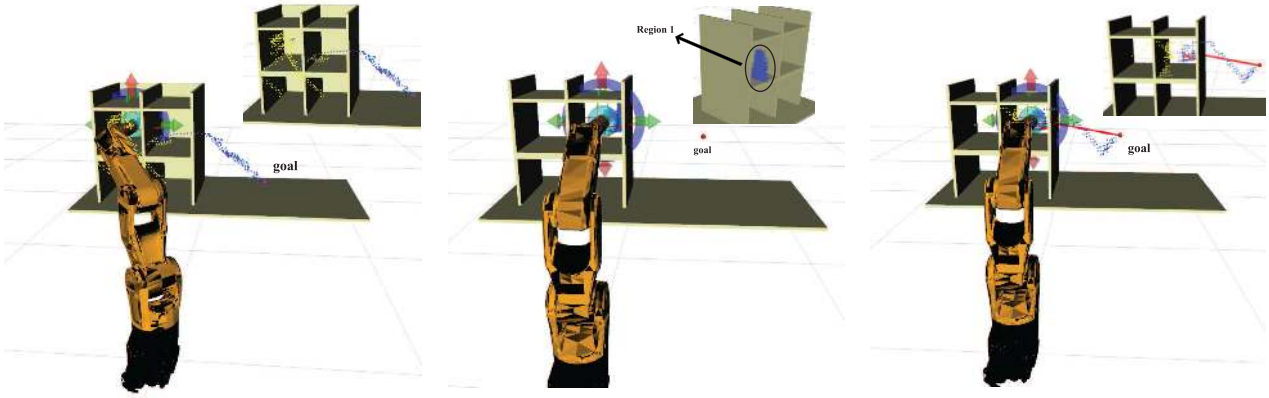
A. MOTION PLANNING ABOVE A DESKTOP

In Figure 5, the first experimental task is to move the manipulator from different positions of the bookshelf to the right side of the desktop. We solve the problem using heuristic search and the first step is to establish heuristic guidance. The red curves in Figure 5 are the corresponding



(a) (b) (c)

FIGURE 5. The consistent heuristic obtained by BFS for different environments or different initial states. The initial states in image (a) and (b) are the same and the environments are different. In contrast, the environments in image (b) and (c) are the same and the initial states are different.



(a) (b) (c)

FIGURE 6. The node expansion results using (a) SMHA* (b) Weighted A* and (c) SD-SMHA*. The blue points are expanded by consistent search and yellow points are expanded by inadmissible searches.

consistent heuristic which is obtained by BFS from initial state. Comparing the guiding curves in Figure 5(a) and Figure 5(b), we find that given the same initial and goal states, the heuristic guidance will change significantly when the experimental scene changes. Obviously the heuristic guidance in Figure 5(b) is not appropriate. Meanwhile, comparing the guiding curves in Figure 5(b) and Figure 5(c), we find that with the same experimental scene, the heuristic guidance will also change significantly when the initial and goal poses change slightly. Therefore, the correctness of consistent heuristic is unpredictable.

For the scenes of Figure 5(a) and (b), we use SMHA* ($\omega_1 = 100, \omega_2 = 1.6$), Weighted A* ($\omega_1 = 100$) and SD-SMHA* ($\omega_1 = 100, \omega_2 = 1.6$) to search separately. Here, we set $\sigma_1 = 100, \sigma_2 = 20, \varepsilon = 20$ and $\varepsilon_{goal} = 60$ for stagnation detection. The node expansion results are shown in Figure 6(a) (b) and (c) respectively. The blue points in Fig 6 indicate the position of the end-effector when the manipulator under the states which are expanded by consistent heuristic. Similarly, the yellow points are extended by the inadmissible heuristics. In Figure 6(a), when the consistent heuristic is appropriate, using SMHA* will add unnecessary inadmissible extensions (the yellow points in Figure 6(a)) since $\omega_2 = 1.6 > 1$. In addition, as can be seen from Figure 6(b), the search is stagnated in Region 1 under the wrong heuristic guidance. Compared to the 2D or 3D space, the complexity of the configuration space of the manipulator is drastically increased, so that the search cannot be separated from the stagnant area for a long time. We add four auxiliary goals which can be seen in Figure 4(b) before the shelf and the search results using SD-SMHA* are shown in Figure 6(c). By adding stagnation detection to both the consistent and inadmissible heuristics, the algorithm can search to the goal quickly.

Figure 7 shows the results of the stagnation detection of the algorithm in the two scenarios of Figure 5(a) and (b). In Figure 7(a), the consistence heuristic is appropriate. Since the constraint of the goal posture is achieved by *adaptive motion primitives*, in order to find a feasible and sub-optimal goal state, the algorithm will fall into a short stagnation near the goal. Therefore, in our implementation, we add a threshold judgment. If the distance between the extended node and the goal are less than a pre-defined threshold ε_{goal}, the stagnation detection will not perform. Here we

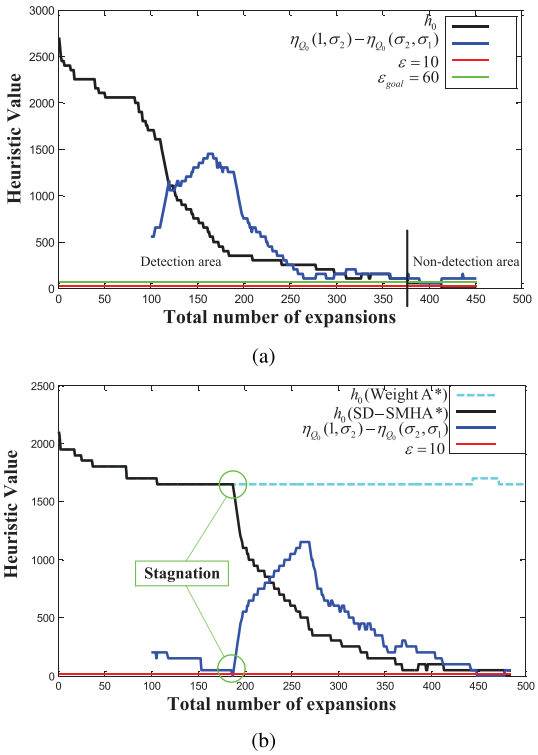


FIGURE 7. The heuristic values of consistent extended nodes and the results of heuristic-based stagnation detection. (a) Node search results in the scenario of Fig 5(a) with an appropriate consistent heuristic. (b) Node search results in the scenario of Fig 5(b) with an error consistent heuristic.

set $\varepsilon_{goal} = 60$ as shown by the green line in Figure 7(a). In Figure 7(b), when the consistence heuristic is error, the search will stagnate with Weighted A* as shown by the light blue dashed line. Using our proposed SD-SMHA* algorithm, the search will effectively converge to the goal by detecting the search stagnation (at the green circle) and introducing inadmissible heuristics.

To confirm the validity of the stagnation detection method and the superiority of the algorithm, we performed 32 experiments using Weighted A*(WA*), SMHA* and SD-SMHA* in the scenario of Figure 5(a) and the consistence heuristic is appropriate in all experiments. The manipulator starts in the four holes of the bookshelf, and the goal pose is randomly selected above the desktop. Four auxiliary goal positions are set in front of the four holes as shown in Figure 4(b). Some of the main parameters are as mentioned above. The planning results are shown in Table 1. It should be noted that the path length is computed by the joint distance through the whole path which is the same as the value of $g(s_{goal})$ in cost function. When the consistence heuristic is appropriate, the success rate of all algorithms above is relatively high and there is no failure during our experiments. In addition, the SD-SMHA* has a low false detection rate of stagnation detection for consistence heuristic and only one occurrence in 33 experiments. Comparing the number of state expansions and planning time, we can find that SMHA* increases unnecessary inadmissible expansions and the planning time increases by 0.02243s

TABLE 1. The Comparison of performance between WA*, SMHA* and SD-SMHA* in the scene of Figure 5(a).

	Path length	False detection rate	States Expanded	Planning time
WA*	**11.6581**	-	**681.429**	**0.04457s**
SMHA*	11.9196	-	821.714	0.067s
SD-SMHA*	11.7281	3.03%	683.429	0.04486s

compared to WA*. Meanwhile, the planning time of SD-SMHA* has only increased by 0.0003s and the increased time is mainly used for stagnation detection. In general, by adjusting the parameters, the stagnation detection method proposed in this paper has low false detection rate and high efficiency. When the consistent heuristic is appropriate, the search efficiency of SD-SMHA* is significantly improved, compared to the original SMHA*.

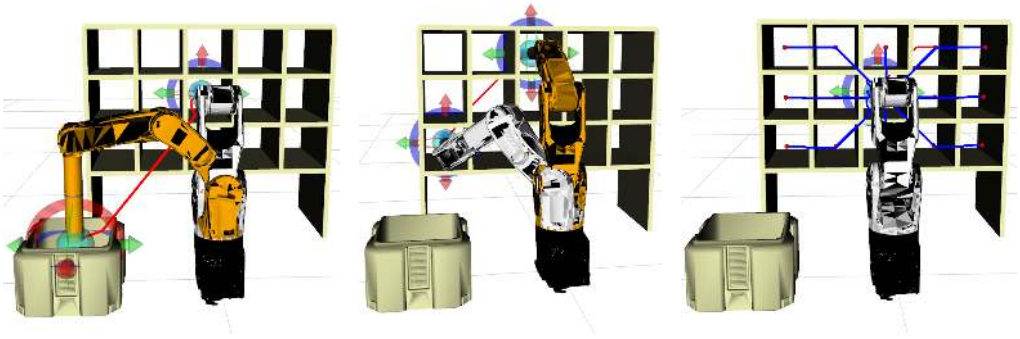
We also experimented in the scene of Figure 5(b) and the results are shown in Table 2. It includes the performance comparing SD-SMHA* with 3 sampling-based algorithms, namely RRT, RRT-Connect and RRT*, in terms of planning time, success rate and solution quality. The settings for the start state and the auxiliary goal positions are similar to the above. We conducted 71 experiments using WA* and SD-SMHA* separately with different initial or goal state. For each of the different initial or goal state, we performed 20 experiments using sampling-based algorithms separately. The results in Table 2 are all averages. In addition, all of the sampling-based algorithms are implemented by the standard OMPL [32]. The results show that the planning time of SD-SMHA* is shorter than the sampling-based algorithms, due to the introduction of correct guidance for the end effector. In addition, taking the path length as a real cost and using the method of graph search, SD-SMHA* is far superior to the sampling-based algorithms in planning path length. Compared to RRT Connect and RRT*, the average path length is shortened by 42.98% and 20.72% respectively. The main disadvantage of SD-SMHA* is that the planning success rate is relatively lower, mainly due to the limitations of discretization for the *static motion primitives* and the inappropriate or insufficient heuristic function which may guide the search to local minima. However, compared with the original WA*, SD-SMHA* greatly improves the planning success rate, by introducing the inadmissible heuristics through the stagnation detection.

B. MOTION PLANNING THROUGH A NARROW SHELF

In Figure 8, motion planning through a narrow shelf is also a common application scenario in real life. In order to prove the superiority of the proposed algorithm, we set two types of tasks. The first is to move the manipulator from any narrow entrance of the shelf to the box on the lower left as shown in Figure 8(a). In Figure 8(b), the second task is to motion

TABLE 2. Comparison between SD-SMHA* and sampling based planners for Denso manipulation in the scene of Figure 5(b).

	RRT	RRT Connect	RRT*(final)	WA*	SD-SMHA*
Success rate	97.2%	**100%**	98.3%	21.13%(15/71)	90.1%(64/71)
Path length	15.9198	16.4278	11.306	9.4406	**9.3657**
States Expanded	-	-	-	**1057.6**	1106.59
Planning time	0.1946s	0.1392s	60s	**0.0678s**	0.0737s

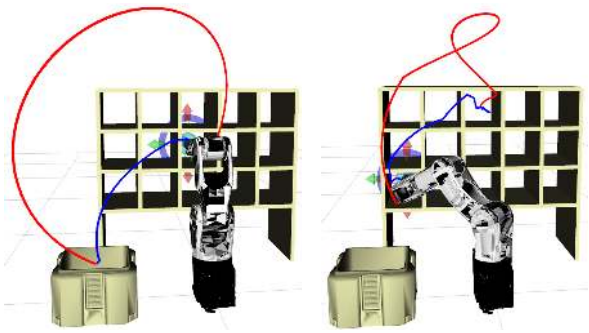


(a) (b) (c)

FIGURE 8. The application scenario with narrow shelf. The image (a) and (b) show the initial state and goal position for task 1 and task 2 respectively. The image (c) shows the 15 auxiliary goal positions we set and the blue curves are the corresponding inadmissible heuristics from the start state.

planning from one entrance of the shelf to another. The red curve in Fig 8 is the consistent heuristic guidance for the end-effector from start state. Obviously for task 1, consistent heuristic is always appropriate. However for task 2, the consistent heuristic is usually inappropriate. The red points at the center of each entrance to the shelf are the 15 auxiliary goal positions and the corresponding inadmissible heuristics from the start state are represented by the blue curves in Figure 8(c). For task1, we randomly select different positions of the shelf as the initial state and the goal is a specified position above the box without posture constraint. In addition, for task 2, the goal is a randomly selected position within the shelf without posture constraint. It should be noted that we have removed the unreachable goal positions near the border. We set $\omega_1 = 100$, $\omega_2 = 1.6$ for search-based algorithms and $\sigma_1 = 150, \sigma_2 = 50, \varepsilon = 20, \varepsilon_{goal} = 100$ for stagnation detection.

We give the end-effector path after post-processing generated by RRTC (red curve) and SD-SMHA* (blue curve) in Figure 9. Obviously, the length of the end-effector path planned by SD-SMHA* is significantly smaller than that planned by RRTC. The specific performance comparison is shown in Table 3. The abbreviations of the performance indicators are represented as follows: SR (Success Rate), PL(Path Length), SE(States Expanded), PT(Planning Time). It should be noted that the path length here refers to the length of the joint path. For task 1, with correct heuristic guidance in narrow spaces, SD-SMHA* is far superior to the sampling-based planners in planning time and planning



(a) (b)

FIGURE 9. The end-effector path generated by RRTC (red curve) and SD-SMHA* (blue curve). The image (a) and (b) correspond to task 1 and task 2 respectively.

path length. Compared with WA* and SD-SMHA*, due to the addition of stagnation detection, SD-SMHA* does not introduce inadmissible heuristics when consistency heuristic is correct, so its parameters are similar to WA* and the stagnation detection process takes little time. However, for SMHA*, due to the addition of unnecessary inadmissible heuristics, the number of states expanded and the planning time increased by 78.44% and 65.12% respectively, compared to SD-SMHA*.

For task 2, we set the maximum planning time to 30s. Since the consistent heuristic is usually inappropriate, the planning success rate of WA* is very low with only 13 successes in 63 experiments (all of the 13 successes are achieved when

TABLE 3. Comparison between searching based planners and sampling based planners in narrow shelf environments.

		RRTC	RRT*	WA*	SMHA*	SD-SMHA*
Task1	SR	**100%**	100%	92.7%(38/41)	97.6%(40/41)	97.6%(40/41)
	PL	18.284	10.9288	5.1361	5.5919	**5.0015**
	SE	-	-	**708.286**	1263.857	716.253
	PT	0.1031s	30s	**0.036**	0.0606	0.03671
Task2	SR	**97.3%**	70.6%	20.6%(13/63)	90.5%(57/63)	90.5%(57/63)
	PL	21.0597	16.254	7.346	6.9104	**6.873**
	SE	-	-	**2354.8**	2639.5	2750.545
	PT	0.707s	30s	**0.1271**	0.1785	0.1821

the consistent heuristic is correct). By introducing the inadmissible heuristics, the planning success rate of SMHA* and SD-SMHA* is greatly improved to 90.5%. Compared to the sample-based planners, SD-SMHA* takes less planning time and joint length of the planned path is much shorter. However, its planning success rate is still lower than RRT Connect. Compared with SMHA* and SD-SMHA*, the performance parameters are close. The planning time of SD-SMHA* is only increased by 2.02% than SMHA*.

In general, SD-SMHA* planner proposed in this paper combines the advantages of WA* and SMHA*. When the consistent heuristic is correct, SD-SMHA* effectively reduces the planning time compared to SMHA*. When the consistent heuristic is incorrect, SD-SMHA* effectively improves the planning success rate compared to WA*. For narrow spaces, SD-SMHA* is superior to sampling-based methods in planning speed and planning trajectory quality. The only drawback is that the planning success rate of SD-SMHA* is still lower than RRT Connect.

VIII. CONCLUSION

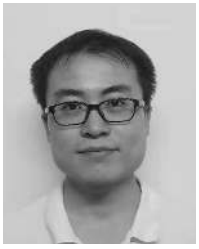
In this paper, we use a heuristic search-based planner to solve the problem of obstacle avoidance path planning for manipulators in narrow space and propose an improved SMHA* algorithm based on stagnation detection. Through introducing multiple inadmissible heuristics, the proposed algorithm can effectively avoid the problem of search stagnation caused by inappropriate consistent heuristic. Meanwhile, through adding stagnation detection to each expansion node, the proposed algorithm can effectively solve the problem that when the consistent heuristic is appropriate, the unnecessary inadmissible heuristics will increases the search burden. In general, the improved algorithm proposed in this paper takes the stagnation detection as a bridge and absorbs the advantages of the original Weighted A* and SMHA* algorithm. In addition, compared with the sampling-based planners, the proposed algorithm is more suitable for cluttered environments with narrow passages, and has more advantages in planning time and planning path length. Finally, there are several directions to improve the current work, such as:

1) attempt of designing special motion primitives for a specific task environment;
2) introducing the idea of Theta* [33] and completing the work of pruning in the search process to further shorten the length of the planning path.

REFERENCES

[1] 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, Aug. 1996.
[2] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, Tech. Rep. TR 98–11, Oct. 1998. [Online]. Available: http://janowiec.cs.iastate.edu/papers/rrt.ps
[3] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Piscataway, NJ, USA, Apr. 2000, pp. 995–1001.
[4] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Las Vegas, NV, USA, Oct. 2003, pp. 1178–1183.
[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
[6] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, St Louis, MO, USA, Mar. 1985, pp. 500–505.
[7] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, Oct. 1992.
[8] C. W. Warren, "Global path planning using artificial potential fields," in *Proc. IEEE Int. Conf. Robot. Autom.*, Scottsdale, AZ, USA, May 1989, pp. 316–321.
[9] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Kobe, Japan, May 2009, pp. 489–494.
[10] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, "Space-time functional gradient optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, May 2014, pp. 6499–6506.
[11] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 4569–4574.
[12] J. Schulman, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Proc. Robot., Sci. Syst.*, Berlin, Germany, 2013, pp. 1–10.
[13] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Stockholm, Sweden, May 2016, pp. 9–15.
[14] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using Gaussian processes and factor graphs," in *Proc. Robot., Sci. Syst.*, Ann Arbor, MI, USA, 2016, pp. 1–9.

[15] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time Gaussian process motion planning via probabilistic inference," *Int. J. Robot. Res.*, vol. 37, no. 11, pp. 1319–1340, Sep. 2018.
[16] P. E. Hart, N. J. 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.
[17] B. J. Cohen, G. Subramania, S. Chitta, and M. Likhachev, "Planning for manipulation with adaptive motion primitives," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 5478–5485.
[18] B. Cohen, S. Chitta, and M. Likhachev, "Single- and dual-arm motion planning with heuristic search," *Int. J. Robot. Res.*, vol. 33, no. 2, pp. 305–320, 2014.
[19] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A," *Int. J. Robot. Res.*, vol. 35, no. 1, pp. 224–243, 2016.
[20] E. Huang, M. Mukadam, Z. Liu, and B. Boots, "Motion planning with graph-based trajectories and Gaussian process inference," in *Proc. IEEE Int. Conf. Robot. Autom.*, Singapore, May 2017, pp. 5591–5598.
[21] K. Yang, S. K. Gan, and S. Sukkarieh, "A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV," *Adv. Robot.*, vol. 27, no. 6, pp. 431–443, Feb. 2013.
[22] M. Du, J. Chen, P. Zhao, H. Liang, Y. Xin, and T. Mei, "An improved RRT-based motion planner for autonomous vehicle in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, May 2014, pp. 4674–4679.
[23] I. Noreen, A. Khan, H. Ryu, N. L. Doh, and Z. Habib, "Optimal path planning in cluttered environment using RRT-AB," *Intell. Service Robot.*, vol. 11, no. 1, pp. 41–52, Jan. 2017.
[24] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robot. Auton. Syst.*, vol. 68, pp. 1–11, Jun. 2015.
[25] Z. Tahir, A. H. Qureshi, Y. Ayaz, and R. Nawaz, "Potentially guided bidirectionalized RRT for fast optimal path planning in cluttered environments," *Robot. Auto. Syst.*, vol. 108, pp. 13–27, Oct. 2018.
[26] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," Comput. Sci. Artif. Intell. Lab, MIT, Cambridge, MA, USA, Tech. Rep. TR-021, Aug. 2013.
[27] A. M. Kabir, B. C. Shah, and S. K. Gupta, "Trajectory planning for manipulators operating in confined workspaces," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Munich, Germany, Aug. 2018, pp. 84–91.
[28] F. Islam, O. Salzman, and M. Likhachev, "Online, interactive user guidance for high-dimensional, constrained motion planning," in *Proc. Int. Joint Conf. Arti. Intell.*, Stockholm, Sweden, Jul. 2018, pp. 4921–4928.
[29] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artif. Intell.*, vol. 1, nos. 3–4, pp. 193–204, 1970.
[30] A. J. Dionne, J. T. Thayer, and W. Ruml, "Deadline-aware search using online measures of behavior," in *Proc. 4th Annu. Symp. Combinat. Search*, Barcelona, Spain, Jul. 2011, pp. 39–46.
[31] P. P. Chakrabarti, S. Ghose, and S. C. Desarkar, "Heuristic search through islands," *Artif. Intell.*, vol. 29, no. 3, pp. 339–347, Sep. 1986.
[32] I. A. ucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.
[33] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta: Any-angle path planning on grids," *J. Artif. Intell. Res.*, vol. 39, pp. 533–579, Oct. 2010.



KAI MI received the B.S. degree from the School of Control Science and Engineering, Shandong University, Jinan, China, in 2015. He is currently pursuing the Ph.D. degree with the Institute of Automation, Chinese Academy of Sciences, Beijing, China. He is also with the University of Chinese Academy of Sciences, Beijing. His research interests include trajectory planning, robotics, and intelligent control systems.

JUN ZHENG was born in 1979. He received the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2009, where he is currently an Associate Professor. His research interests include intelligent robots, servo control, and motion control systems.

YUNKUAN WANG was born in 1966. He received the M.Sc. degree in industrial automation from the Harbin Institute of Technology, Harbin, China, in 1992. He is currently a Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include intelligent robots, visual servo, and complex control systems.

JIANHUA HU was born in 1987. He received the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2014, where he is currently an Associate Professor. His research interests include high-speed visual recognition, digital printing, and intelligent robots.