Long-Horizon Motion Planning for Autonomous Vehicle Parking Incorporating Incomplete Map Information

Siyu Dai and Yebin Wang

***Abstract*— This paper presents a hierarchical motion planning approach that can provide real-time parking plans for autonomous vehicles with limited memory. Through combining a high-level route planner that searches for collision-free routes and a low level motion planner that considers vehicle dynamics, our approach generates smooth trajectories with reasonable parking behaviors rapidly with very low memory consumption. In addition, this hierarchical approach also allows for online path repairing and replanning when newly detected obstacles that were not indicated on the offline map obstruct the original planned trajectory. It employs a fast clearance checking procedure to obtain a practical indicator of repairability as well as heuristic guidance for rapid trajectory repairing, and utilizes the high-level route planner to conduct real-time replanning when trajectory repairing is deemed to be difficult. Performance analysis on parking tasks in simulation environments demonstrates the advantage of the proposed approach in terms of both trajectory quality and planning time.**

I. Introduction

Although semi-autonomous parking assistance systems are already popular in the commercial vehicle market, fully autonomous valet parking [1], [2] remains challenging. In order to allow human drivers to simply drop off or request the vehicles from outside of parking lots, the autonomous vehicle parking system must be able to conduct both long-range navigation inside cluttered environments as well as short-range maneuvering around narrow parking spaces. This requires the seamless integration of multiple intelligent components including perception, localization, task planning, motion planning and vehicle control. In this paper, we focus mainly on the motion planning component of an autonomous valet parking system that searches for collision-free and dynamically feasible trajectories given known start and goal poses as well as the parking lot map.

General motion planning algorithms [3]–[9] lack the ability of rapidly producing dynamically feasible and collision-free solutions for navigation through cluttered environments with narrow passages in a memory-efficient manner, since it often requires complicated driving maneuvers and safety guarantees. Existing autonomous vehicle parking systems often inherit the above issues when they directly employ general motion planners, and many motion planners specialized for autonomous parking still fail to integrate short-horizon planning with long-horizon planning or cannot incorporate online path repairing upon new obstacle information [2], [10]–[15]. Wang [16] proposed Bi-Directional A-Search Guided Tree (BIAGT) to overcome the low computational efficiency of existing search-based motion planners and successfully demonstrated the advantage of this fast kinodynamic motion planner on autonomous vehicle parking tasks. Although BIAGT improves heuristic estimation through a two-tree structure, its effectiveness in terms of integrating vehicle dynamics and obstacle information is still limited especially when the planning horizon is long. Therefore, in this paper, we present *Route-Planning Bi-Directional A-Search Guided Tree (RP-BIAGT)*, a hierarchical search-based motion planning framework that achieves fast heuristic estimation while maintaining both vehicle dynamics and obstacle information. In addition to allowing for rapid long-horizon motion planning with low memory consumption, this hierarchical framework also provides a convenient interface for enforcing natural driving behaviors and incorporating new obstacle information during execution. With a fast geometric clearance checking procedure, RP-BIAGT can intelligently decide whether to follow the original route and repair the trajectory or to replan. We evaluate the performance of RP-BIAGT on a variety of parking tasks in simulation environments and demonstrate its advantages in terms of initial planning, trajectory repairing and replanning.

II. Related Work

Existing motion planning approaches mostly fall into three categories: search-based [17]–[19], sampling-based [20]–[22] and optimization-based [23]–[26]. Sampling-based motion planners are especially popular in humanoid robotic tasks thanks to their effectiveness in high-dimensional environments [27]–[29], but inevitably raise concerns in risk-sensitive tasks such as autonomous driving due to their non-deterministic nature. In addition, their inefficiency caused by larger numbers of collision checks as well as the suboptimality of their solution trajectories significantly constrains their application in real-time driving tasks. Optimization-based motion planners, on the other hand, operate on the space of trajectories and naturally incorporate the non-holonomic vehicle dynamics. However, since they conduct local search instead of global search and inevitably suffer from suboptimality caused by high-cost local optima, they are typically combined with other global planners instead of operating alone [26], [30]–[32]. Variants of search-based motion planners are widely used on autonomous vehicles [3], [33]–[39], but how to conduct long-range navigation through cluttered environments in a time and memory efficient manner remains challenging. The approach proposed in this paper

S. Dai is with MIT, Cambridge, MA 02139, USA. This work was done while she was a research intern at Mitsubishi Electric Research Laboratories. (email: sylviad@mit.edu).

Y. Wang is with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA (email: yebinwang@ieee.org).

overcomes the disadvantages of existing search-based motion planners through the concept of driving modes as well as a hierarchical framework, and significantly improves the time and memory efficiency for motion planning in long-horizon autonomous valet parking tasks.

Real-time trajectory repairing has been approached from various different aspects, including online heuristic update [40]–[42], pruning and reconnecting sampling-based search structures [43]–[46], and spline-based kinodynamic search [47], [48]. Heuristic update methods are designed for graph-based structures and are not directly applicable in our tree-based search structure in RP-BIAGT, and the tree pruning and reconnecting methods are less desirable in the memory-constrained scenarios we are considering. The repairing and replanning procedure proposed in this paper is inspired by spline-based repairing methods, but differs significantly in that, instead of directly conducting kinodynamic search, we apply a hierarchical framework which first conduct geometric shifting and then search for dynamically feasible trajectories with the heuristic guidance from the shifting. This not only improves the computational efficiency but also provides a convenient criterion to decide between trajectory repairing and replanning.

III. Preliminaries

A. Problem Statement

We consider the planning problem with vehicle dynamics:

$$\dot{X} = f(X) + g(X)u, \tag{1}$$

where $X \in \mathcal{X} \subset \mathbb{R}^{n_x}$ is the state and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input. A collision-free configuration space $\mathcal{C}_{free} \subset \mathbb{R}^{n_c}$ is defined as the set of configurations at which the vehicle has no intersection with obstacles in the environment. An *admissible trajectory* $\mathcal{X}_t$ is a solution of system (1) with given initial and final conditions and $u \in \mathcal{U}$, and a *feasible trajectory* is an admissible trajectory that is also collision-free. The motion planning problem considered in this paper is defined as follows:

Problem 3.1: Given an initial configuration $X_0 \in \mathcal{C}_{free}$, a goal configuration $X_f \in \mathcal{C}_{free}$, and system (1), find a feasible trajectory $\mathcal{P}_t$ which

(I) starts at X_0 and ends at X_f, while satisfying (1); and
(II) lies in the collision-free configuration space $\mathcal{C}_{free}$.

B. Bi-Directional A-Search Guided Tree (BIAGT)

As a variant of A*-based algorithms, Bi-Directional A-Search Guided Tree (BIAGT) [16] is proposed to efficiently provide feasible solutions for autonomous vehicle parking tasks. We define a tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ as a union of a node set $\mathcal{V} \subset \mathcal{C}_{free}$ and an edge set $\mathcal{E}$, where $E(X_i, X_j) \in \mathcal{E}$ represents a feasible trajectory between X_i and X_j. Let $\mathcal{M}$ denote a finite set of motion primitives pre-computed through available control actions, and V_{max} the maximum number of nodes allowed. As described in Algorithm 1, BIAGT constructs a start tree $\mathcal{T}_s$ and a goal tree $\mathcal{T}_g$ rooted from X_0 and X_f respectively, and expands them according to a cost function $F(\cdot)$ which sums up the heuristic value $h(\cdot)$ and the arrival cost $g(\cdot)$. The heuristics in BIAGT are calculated based on the Reeds-Shepp (RS) path length [49] towards the tree's corresponding goal without considering obstacles. If either tree gets close to their corresponding goal or the distance between two trees is smaller than or equal to a threshold ϵ, then BIAGT will connect the two trees with a dynamically feasible path. If the connection is successful, all the parents of the connection nodes are added to the other tree to form a feasible solution trajectory and the motion planning is viewed as successful. If the total number of nodes on both tree reaches V_{max} but a solution trajectory is still not found, then the motion planning has failed.

Algorithm 1: BIAGT

input $\mathcal{C}_{free}, X_0, X_f, V_{max}, \mathcal{M}, \epsilon$
$\mathcal{T}_s \leftarrow (X_0, \emptyset), \mathcal{T}_g \leftarrow (X_f, \emptyset)$
$F(X_0) \leftarrow g(X_0) + h(X_0), F(X_f) \leftarrow g(X_f) + h(X_f)$
$Q_s \leftarrow (X_0, F(X_0), Q_g \leftarrow (X_f, F(X_f)$
$k \leftarrow 2$, $success \leftarrow$ **false**
while $k \leq V_{max}$ **and not** $success$ **do**
  $X^{\mathrm{s}}_{\mathrm{best}} = Q_s$.Pop **where** $F(X^{\mathrm{s}}_{\mathrm{best}}) \leq F(X), \forall X \in Q_s$
  $X^{\mathrm{g}}_{\mathrm{best}} = Q_g$.Pop **where** $F(X^{\mathrm{g}}_{\mathrm{best}}) \leq F(X), \forall X \in Q_g$
  if $d(X^{\mathrm{s}}_{\mathrm{best}}, X_f) \leq \epsilon$ **or** $d(X^{\mathrm{g}}_{\mathrm{best}}, X_0) \leq \epsilon$ **then**
    $success \leftarrow$ `true`
  else if $dist(\mathcal{T}_s, \mathcal{T}_g) \leq \epsilon$ **then** `Connect`$(\mathcal{T}_s, \mathcal{T}_g)$;
  else
    $(success, n_s) \leftarrow$ `Expand`$(\mathcal{T}_s, \mathcal{C}_{free}, X^{\mathrm{s}}_{\mathrm{best}}, \mathcal{M})$
    Compute F for new nodes and append to Q_s
    if not $success$ **then**
      $(success, n_g) \leftarrow$ `Expand`$(\mathcal{T}_g, \mathcal{C}_{free}, X^{\mathrm{g}}_{\mathrm{best}}, \mathcal{M})$
      Compute F for new nodes and append to Q_g
    $k \leftarrow k + n_s + n_g$
$\mathcal{P}_t =$ `ReturnFeasiblePath`$(\mathcal{T}_s, \mathcal{T}_g, success)$
return $\mathcal{P}_t$

IV. Approach

Despite BIAGT's capability of solving short-range autonomous parking problems demonstrated in [16], it inevitably suffers from long planning time and high memory consumption when the task is long-range and when updated obstacle information received from sensors invalidates the original trajectory. In order to address these issues, in this paper we propose Route-Planning Bi-Directional A-Search Guided Tree (RP-BIAGT) which extends the original BIAGT through three contributions: 1) a hierarchical framework that provides significant performance improvement in terms of planning time, trajectory smoothness and memory consumption (Section IV-A); 2) a trajectory-repairing procedure upon updated map information which uses geometric configuration shifting to provide heuristic guidance as well as repairing difficulty estimation (Section IV-B); 3) a fast replanning approach that searches for new feasible routes and trajectories when repairing is determined to be difficult (Section IV-C). Fig. 1 and Algorithm 2 illustrates RP-BIAGT's overall framework and algorithmic details respectively.

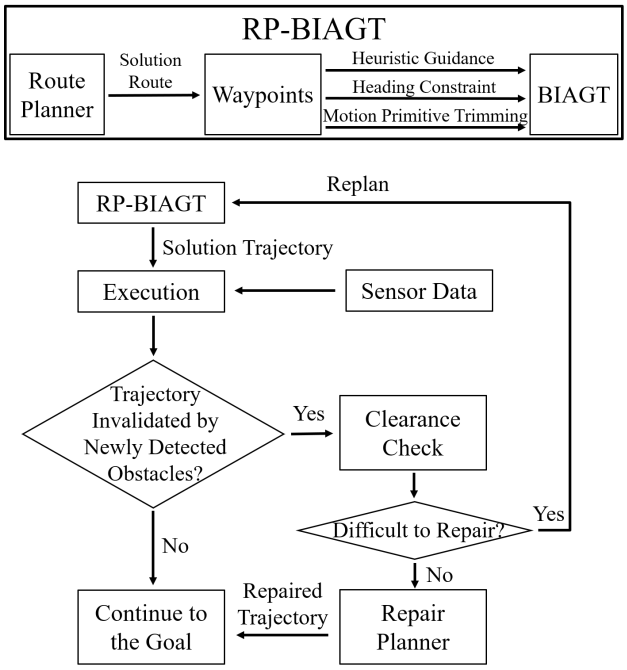
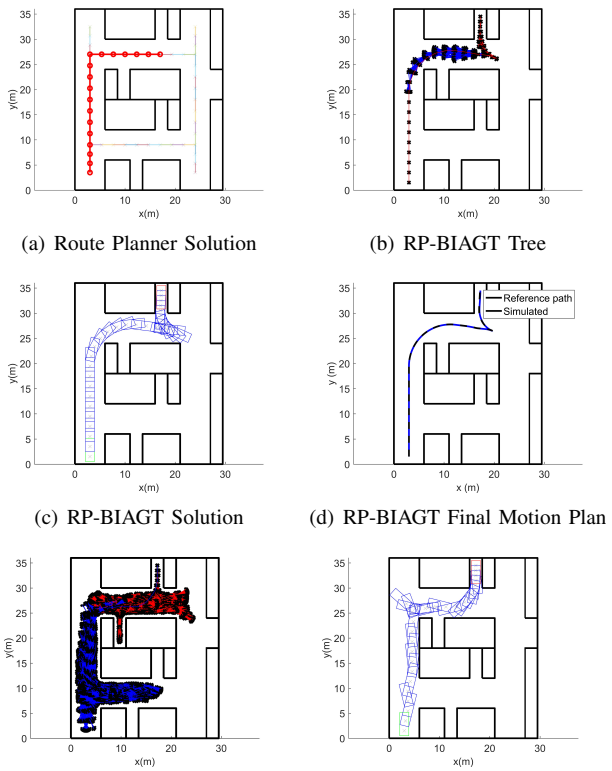


Fig. 1: RP-BIAGT System Diagram

Algorithm 2: RP-BIAGT

input $\mathcal{C}_{free}, X_0, X_f, V_{max}, \mathcal{M}, \epsilon, \eta$
$\mathcal{T}_s \leftarrow (X_0, \emptyset), \mathcal{T}_g \leftarrow (X_f, \emptyset)$
$RP = \texttt{RoutePlanning}(\mathcal{C}_{free}, X_0, X_f)$
$h(X_0) = \texttt{RPHeuristic}(RP, X_0, X_f)$
$h(X_f) = \texttt{RPHeuristic}(RP, X_f, X_0)$
$F(X_0) \leftarrow g(X_0) + h(X_0), F(X_f) \leftarrow g(X_f) + h(X_f)$
$Q_s \leftarrow (X_0, F(X_0), Q_g \leftarrow (X_f, F(X_f)$
$k \leftarrow 2$, $success \leftarrow$ **false**
while $k \leq V_{max}$ **and not** $success$ **do**
  $X^{\mathrm{s}}_{\mathrm{best}} = Q_s.\texttt{Pop}$ **where** $F(X^{\mathrm{s}}_{\mathrm{best}}) \leq F(X), \forall X \in Q_s$
  $X^{\mathrm{g}}_{\mathrm{best}} = Q_g.\texttt{Pop}$ **where** $F(X^{\mathrm{g}}_{\mathrm{best}}) \leq F(X), \forall X \in Q_g$
  if $d(X^{\mathrm{s}}_{\mathrm{best}}, X_f) \leq \epsilon$ **or** $d(X^{\mathrm{g}}_{\mathrm{best}}, X_0) \leq \epsilon$ **then**
    $success \leftarrow$ **true**
  else if $dist(\mathcal{T}_s, \mathcal{T}_g) \leq \epsilon$ **then** $\texttt{Connect}(\mathcal{T}_{\mathrm{s}}, \mathcal{T}_{\mathrm{g}})$;
  else
    Trim $\mathcal{M}$ **if** $\texttt{HeadingAlign}(X^{\mathrm{s}}_{\mathrm{best}}, RP)$
    $(success, n_s) \leftarrow \texttt{Expand}(\mathcal{T}_s, \mathcal{C}_{free}, X^{\mathrm{s}}_{\mathrm{best}}, \mathcal{M})$
    Compute F for new nodes and append to Q_s
    if not $success$ **then**
      Trim $\mathcal{M}$ **if** $\texttt{HeadingAlign}(X^{\mathrm{g}}_{\mathrm{best}}, RP)$
      $(success, n_g) \leftarrow \texttt{Expand}(\mathcal{T}_g, \mathcal{C}_{free}, X^{\mathrm{g}}_{\mathrm{best}}, \mathcal{M})$
      Compute F for new nodes and append to Q_g
    $k \leftarrow k + n_s + n_g$
$\mathcal{P}_t = \texttt{ReturnFeasiblePath}(\mathcal{T}_s, \mathcal{T}_g, success)$
Execute $\mathcal{P}_t$ until new obstacles detected
if $\mathcal{P}_t$ in collision **then**
  $\mathcal{C}_c \leftarrow$ Nodes in collision on $\mathcal{P}_t$
  $(\mathcal{C}_s, Clearance_{\max}) \leftarrow \texttt{LateralShift}(\mathcal{C}_{\mathrm{c}})$
  if not all $\mathcal{C}_s$ **or** $Clearance_{\max} \leq \eta$ **then** Replan;
  else
    $(X_0^{new}, X_f^{new}, \mathcal{P}_t^{C\text{-}free}) \leftarrow \texttt{ResetTask}(\mathcal{C}_{\mathrm{c}}, \mathcal{P}_{\mathrm{t}})$
    Repeat line $2-24$ with $X_0^{new}, X_f^{new} \rightarrow \mathcal{P}_t^{new}$
    $\mathcal{P}_t \leftarrow \texttt{PathConnect}(\mathcal{P}_{\mathrm{t}}^{\mathrm{new}}, \mathcal{P}_{\mathrm{t}}^{\mathrm{C\text{-}free}})$
return $\mathcal{P}_t$



(a) Route Planner Solution (b) RP-BIAGT Tree

(c) RP-BIAGT Solution (d) RP-BIAGT Final Motion Plan

(e) BIAGT Tree (f) BIAGT Solution

Fig. 2: Comparison of RP-BIAGT and BIAGT on Demo 14. The green box and red box in (c) show the start and goal pose for the vehicle respectively, and the blue lines and red lines in (b) represent the start tree and the goal tree respectively.

A. Route Planning

A fundamental shortcoming of BIAGT is that its heuristics only consider vehicle dynamics and not obstacle information, which may result in tree expansions into dead ends when feasible trajectories are long and winding. To tackle this issue, we propose a hierarchical framework that combines BIAGT with a high-level route planner, as illustrated through an example parking task in Fig. 2. The route planner first generates a directed graph based on the parking lot map and the traffic direction information and then uses A* algorithm to search for the shortest path between start and goal nodes on the graph, as shown in Fig. 2(a). We refer to this shortest path as the *route planner solution route*, and the nodes on this shortest path the *waypoints*. These waypoints serve three roles in RP-BIAGT: 1) the route length from a waypoint to its corresponding goal $h_{waypoint}$ is used when the low level BIAGT computes heuristics; 2) the heading of the waypoint closest to the parking space guides BIAGT to get out in the correct direction; 3) the heading of the waypoints on straight driveways are utilized to trim available motion primitives in order to avoid unnecessary steering motions and to reduce the number of nodes required to find a feasible trajectory.

a) Heuristic Guidance: Instead of computing heuristics based on the RS path length to the goal, RP-BIAGT searches for the k-nearest waypoints based on Euclidean distance to the current node being expanded and computes a set of

heuristics h_i that add up the to-go-cost from each of the nearest waypoints $h_{waypoint}$ and their RS distance to the current node RS(node,waypoint). The minimum of this set of heuristic values is used as the actual heuristic of the current node, as shown in Equation 2.

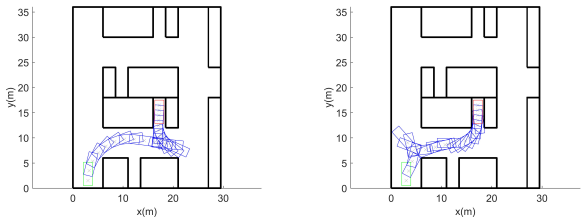
$$\begin{aligned} h_i &= RS(\text{node,waypoint}) + h_{waypoint},\ i = 1, \ldots, k \\ h &= \min_i h_i \end{aligned} \tag{2}$$

b) Heading Constraint: Without human knowledge baked in, motion planners often come up with trajectories that are optimal in terms of path length but violate human logic. As shown in Fig. 2(f) and 3(b), BIAGT solution instructs the vehicle to drive backwards since very far away from the parking space, which might cause confusion in parking lots that accommodate both human drivers and autonomous vehicles. In contrast, since the route planner solution routes are constrained by the traffic direction, RP-BIAGT provides a convenient interface for enforcing natural parking behaviors. For the trees growing out of parking spaces, RP-BIAGT defines a heading indicator that only turns true when the heading difference between the current node and the first waypoint outside of the parking space is smaller than a threshold θ. Nodes on these trees will only use the first waypoint outside of the parking space for heuristic guidance instead of the k-nearest waypoints until the heading indicator turns true, so that fixing heading is prioritized.

c) Motion Primitive Trimming: A crucial drawback of BIAGT is that the number of nodes it takes to solve long-horizon parking tasks often exceeds the memory constraint of practical autonomous vehicles. In addition, although fine grids of motion primitives help maneuvering the vehicle around parking spaces, they could cause the trajectory in long driveways to be unnecessarily winding. Therefore, in RP-BIAGT we introduce three *driving modes*, Parallel, Normal and Navigation, to distinguish and provide suitable motion primitives for parallel parking, normal parking and forward driving. Both Parallel and Normal modes allow forward and backward driving, but Parallel has finer grids for motion primitives. Navigation mode only allows forward driving and uses longer arcs for motion primitives. Additionally, RP-BIAGT also provides a motion primitive trimming procedure for Navigation mode which first checks the heading of the current node with k-nearest waypoints and then trim all the steering primitives if the heading difference between the node and all waypoints ahead is within a threshold ψ. This not only avoids unnecessary steering and enhances trajectory smoothness, but also helps restricting the total number of nodes within a practical limit to satisfy the memory constraint. If we compare Fig. 2(b)-(c) with Fig. 2(e)-(f), we can see that RP-BIAGT utilizes tree nodes much more efficiently than BIAGT and also generates much smoother trajectories.

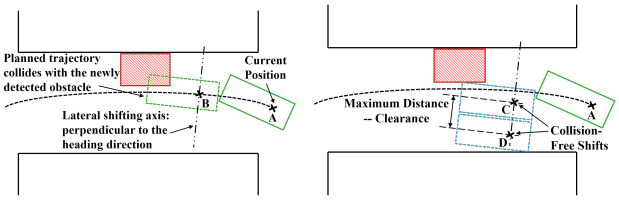
B. Trajectory Repairing

Although static parking lot map information can often be obtained offline, real-time obstacle detection and path repairing is essential for any robust autonomous parking



(a) RP-BIAGT Solution (b) BIAGT Solution

Fig. 3: Comparison of RP-BIAGT and BIAGT on Demo 12.



(a) (b)

Fig. 4: Illustration of the Repairing Criterion

solution to accommodate small, temporary road-blocks such as traffic cones or misplaced shopping carts. Ultimately, we aim at integrating RP-BIAGT with sensing components for real-time obstacle detection and parking space searching, but in this paper, we assume updated map information can be obtained and focus solely on motion planning without higher level decisions involving changes to the goal. When newly detected obstacles invalidate the original trajectory, drivers need to choose between following the same route (*repairing*) and finding a different route that reaches the same goal (*replanning*). Repairing is often preferred considering the execution time and energy consumption, but when the driveway becomes very narrow due to large or inconveniently placed new obstacles, long planning time and high collision risk makes repairing undesirable even if it is feasible. Therefore, it is important to maintain a "repairing criterion" to intelligently decide between repairing and replanning.

In this paper, we propose a geometric *clearance check* procedure to provide a rapidly computable repairing criterion, as illustrated with an example in Fig. 4. A new obstacle is detected when the vehicle drives to configuration A, hence it searches for all the nodes along the original trajectory that collide with the new obstacle and finds the corresponding lateral axis perpendicular to the heading direction for each node. It then shifts the configuration along the lateral axis in both directions in search of collision-free shifts, and will rotate the axis to repeat this procedure if none was found. We define the maximum distance between collision-free shifts (in Fig. 4 the distance between configuration C and D) as the *clearance*. The RP-BIAGT repair planner repeats this procedure for all nodes in collision. If for any node collision-free shifts don't exist or the clearance is smaller than a threshold η, then the repairing criterion is not met and RP-BIAGT will go directly to replanning. Otherwise, RP-BIAGT would repair the section of the original trajectory that collides with the new obstacles and connect it with the collision-

TABLE I: Clearance and Repairability for Demo 25

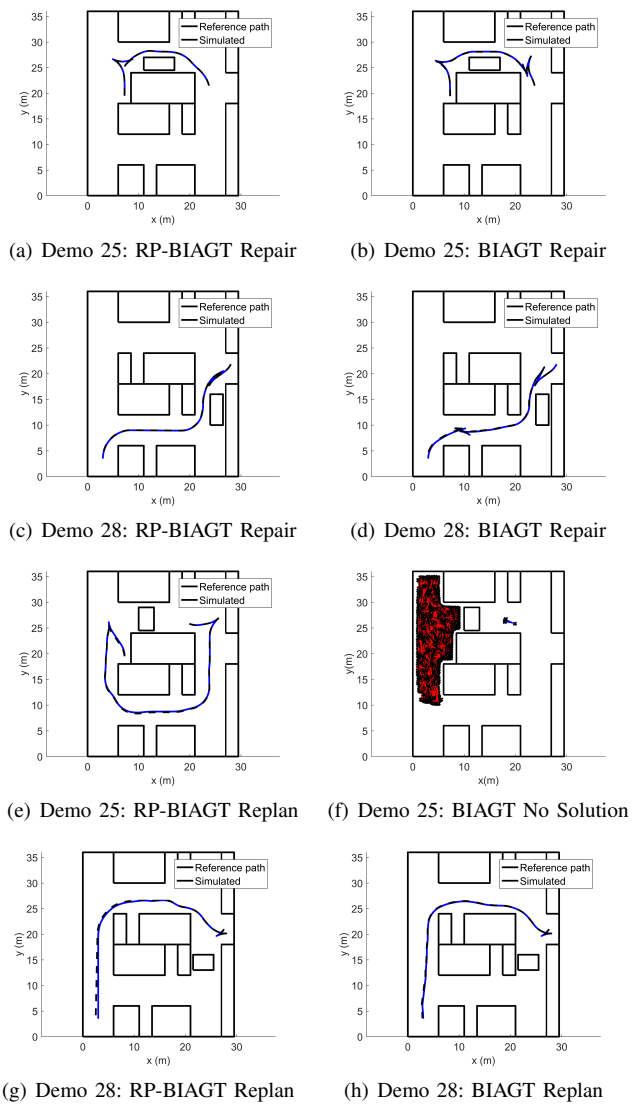
Obstacle Width	Theoretical Clearance[1]	# of Nodes without Shifts[2]		Minimum Clearance		Average Clearance[3]		# of Nodes to Repair[4]	
		$l = 6$	$l = 7$	$l = 6$	$l = 7$	$l = 6$	$l = 7$	$l = 6$	$l = 7$
2.5	1.19	0	0	0.60	0.60	1.43	1.38	326	1306
2.6	1.09	0	0	0.45	0.45	1.31	1.26	509	6276
2.7	0.99	0	0	0.30	0.30	1.23	1.16	1238	5146
2.8	0.89	0	0	0.30	0.30	1.18	1.14	7226	13561
2.9	0.79	0	0	0.15	0.15	1.06	1.01	5199	12782
3	0.69	0	0	0.00	0.00	0.99	0.92	14603	12639
3.1	0.59	0	0	0.00	0.00	0.47	0.42	13683	17839
3.2	0.49	0	0	0.30	0.28	0.44	0.37	15233	24272
3.3	0.39	0	0	0.15	0.15	0.34	0.30	30000	24784
3.4	0.29	0	0	0.00	0.00	0.27	0.22	30000	30000
3.5	0.19	0	0	0.00	0.00	0.19	0.12	30000	30000
3.6	0.09	3	3	0.00	0.00	0.17	0.12	30000	30000
3.7	-0.01	4	5	0.00	0.00	0.15	0.12	30000	30000

[1] Theoretical clearance is driveway width (6 m) minus obstacle width, minus the gap between obstacle and driveway border (0.5 m), and minus vehicle width (1.81 m). In this table, obstacle width, obstacle length (l) and clearance are all in meters (m).
[2] Among the nodes on the original trajectory that are in collision, how many of them don't have collision-free shifts.
[3] Minimum clearance and average clearance are the minimum and average of the clearance values respectively for all the collision nodes on the original trajectory reported by RP-BIAGT.
[4] Maximum number of nodes allowed is 30000.

free sections. In practice, it is usually infeasible to directly connect the immediate neighbors of the collision nodes due to vehicle dynamics, thus in RP-BIAGT, we propose to start repairing m nodes away from the collision nodes, where m is a customizable parameter that depends on the vehicle size and the arc lengths of motion primitives.

In order to analyze the relationship between clearance and repairing difficulty, we experiment on two demo tasks with additional obstacles of different sizes. For brevity, we only show the results for Demo 25 with two sets of lengths ($l = 6$ m and $l = 7$ m) and thirteen sets of widths in Table I. In all experiments in this paper, we conduct discrete shifts with step size 0.15 m and allow for $\pm 10°/20°$heading changes when lateral shifting fails to find collision-free configurations. From Table I we can see that for Demo 25, when the average clearance is lower than 1.2 m, the number of nodes for repairing the trajectory starts to exceeds 5000, the ideal threshold for memory consumption for the vehicles we target at. When the clearance drops below 0.3 m or when some nodes can't find collision-free shifts, it starts to become infeasible to repair within 30000 nodes. It is also noted that as obstacle grows wider, the minimum clearance among all collision nodes first drops to zero and then increases before it drops to zero again. This is because in our experiments, when RP-BIAGT can find collision-free shifts without changing heading, it will skip the heading rotation step and report the clearance value even if there's only one collision-free shift. However, it might find more collision-free shifts and show a non-zero clearance when it rotates the heading direction. A similar trend is also observed in experiments for Demo 28.

In addition to providing the clearance for the repairing difficulty test, these geometric shifts also provide the repair planner with heuristic guidance. When the repairing criteria



(a) Demo 25: RP-BIAGT Repair (b) Demo 25: BIAGT Repair

(c) Demo 28: RP-BIAGT Repair (d) Demo 28: BIAGT Repair

(e) Demo 25: RP-BIAGT Replan (f) Demo 25: BIAGT No Solution

(g) Demo 28: RP-BIAGT Replan (h) Demo 28: BIAGT Replan

Fig. 5: Comparison between RP-BIAGT and BIAGT on trajectory repairing and replanning tasks. (a)-(d) show demo 25 and 28 with newly detected obstacles when trajectories can be easily repaired, and (e)-(h) demonstrate cases where repairing is difficult and the replanning procedure is triggered. In demo 25, BIAGT is unable to find a new plan within 30000 nodes, thus the trees are shown in (f) instead of the solution trajectory.

are satisfied, the median of the collision-free shifts for each collision node are used in Equation 2 instead of the original route planner waypoints for heuristic calculation. Fig. 5(a)-(d) qualitatively compare the performance of the RP-BIAGT repair planner and BIAGT, and we can see that RP-BIAGT is able to provide smoother trajectories for path repairing.

C. Replanning

When RP-BIAGT determines that it is difficult to repair the original trajectory, it will remove the directed graph edges that collide with the new obstacles and use route planner to find a new route. It then follows a similar procedure as described in Section IV-A to find a dynamically feasible trajectory along this route. The main difference in

the replanning procedure is that we use heading constraints for all start trees even if they are not growing out of parking spaces. This is because rerouting typically involves turning to the opposite of the original driving direction, which is especially challenging for heuristic-driven motion planners. Without heading constraints, these planners would often instruct the vehicle to drive backwards all the way to the goal, which is optimal from the path length point of view but very undesirable considering human logic. Therefore, we use heading constraints described in Section IV-A.0.b on all start trees in addition to the goal trees that grow out of parking spaces to ensure natural driving behaviors.

V. Empirical Evaluation

We test RP-BIAGT's initial planning, repairing and replanning performance in various parking tasks in simulation environments, and the comparison with BIAGT in terms of planning time and node number is shown in Table II and III. All the experiments are run on a 6-core Intel i7 3.7GHz desktop with Matlab R2020a.

RP-BIAGT's initial planning performance is tested on 19 driving tasks in two different parking lots. All tasks are between one parking space and one exit, and are categorized based on the distance between start and goal and the driving direction. The tasks with their goal inside parking spaces are called Park-In tasks and the ones with goals at exits are called Drive-Out tasks. All perpendicular parking spaces require the front of the vehicle to be at the opening of the parking space, and the Park-In tasks are much more challenging than the Drive-Out tasks since they involve changes of the heading direction at the parking space. From Table II we can see that in most demos, especially the long-range ones, RP-BIAGT uses significantly fewer nodes to find a solution compared to BIAGT. Although RP-BIAGT takes more time per node because it goes through the hierarchical procedure of finding k-nearest waypoints and computing a set of heuristics, it almost always takes less time for the same task due to the significant reduction of node numbers.

We evaluate RP-BIAGT's performance in terms of repairing and replanning on four of the demo tasks by adding new obstacles after the vehicle drives out. The testing vehicle is 1.81 m wide, and the repairing performance shown in Table III is tested on tasks with a 3 m passage after adding the new obstacle, thus trajectory repairing should be feasible. The replanning performance shown in Table III is tested on tasks with larger new obstacles where the passage becomes 1 - 1.5 m wide and trajectory repairing is infeasible. From Table III we can see that BIAGT is rarely able to find a new trajectory for replanning within 30000 nodes, and compared to RP-BIAGT, it also tend to take a larger number of nodes to repair the trajectory in cases with smaller new obstacles.

TABLE II: RP-BIAGT and BIAGT Comparison

Demo No.	Demo Type		RP-BIAGT		BIAGT	
	Long or Short Range	Park-In or Drive-Out	Planning Time (s)	# of Nodes	Planning Time (s)	# of Nodes
11	Short	In	0.81	1129	0.88	1417
12	Short	In	0.55	637	0.65	1229
13	Long	In	0.44	446	8.76	10720
14	Long	In	0.56	660	11.04	13132
15	Long	In	0.95	912	1.66	1562
16	Long	In	0.99	1145	3.10	4557
17	Long	In	0.79	953	35.38	29473
18	Long	In	0.42	460	8.44	9781
19	Long	Out	0.19	194	2.56	4276
20	Long	In	0.98	702	1.97	1446
21	Long	Out	0.34	424	36.82	30000*
22	Long	Out	0.39	339	32.16	30000*
23	Short	In	0.92	915	0.99	1664
24	Long	In	0.51	593	0.19	373
25	Long	In	1.69	2092	2.01	3637
26	Short	Out	0.39	599	0.20	351
27	Long	Out	0.43	597	0.89	1729
28	Long	Out	0.42	427	2.42	4197
29	Long	In	0.78	860	2.01	3757

* The maximum number of nodes allowed is 30000, and BIAGT failed to find feasible trajectories within 30000 nodes in these cases.

TABLE III: Path Repairing and Replanning Comparison

Demo No.**	Repair				Replan			
	RP-BIAGT		BIAGT		RP-BIAGT		BIAGT	
	Time (s)	# of Nodes	Time (s)	# of Nodes	Time (s)	# of Nodes	Time (s)	# of Nodes
14	0.96	900	0.69	1234	1.39	1598	35.52	30000*
25	0.37	326	2.31	3510	0.66	542	35.66	30000*
27	0.13	147	0.29	287	0.35	352	39.76	30000*
28	0.41	339	1.03	1049	0.31	324	14.38	13763

* The maximum number of nodes allowed is 30000, and BIAGT failed to find feasible trajectories within 30000 nodes in these cases.

** For the same demo, the repairing task and the replanning task have the same basic map but different new obstacles. The new obstacles in the replanning tasks are relatively larger so that the repairing criterion will inform RP-BIAGT to go directly to the replanning procedure instead of attempting to repair.

VI. Discussion

This paper presents RP-BIAGT, an efficient search-based motion planning approach that can rapidly produce dynamically-feasible and memory-efficient solutions for autonomous vehicle valet parking scenarios. Through a hierarchical framework that combines a high-level map-aware route planner and a low-level dynamics-aware bi-directional search tree, RP-BIAGT incorporates information on driving direction, environment obstacles and vehicle dynamics in its heuristics in order to achieve time and memory efficiency. In addition to providing a convenient interface for incorporating human knowledge and enforcing natural driving behaviors through its hierarchical framework, RP-BIAGT also accommodates new obstacles detected during execution with a fast geometric shifting procedure that estimates the clearance and decides between repairing and replanning. When conducting trajectory repairing, the geometric shifts can also replace the original route planner solution as the new heuristic guidance to facilitate rapid repairing. In future research, we hope to enhance RP-BIAGT's performance by improving the time-consuming components through supervised learning, including tree connection and repairability test. Furthermore, we also hope to replace the discrete-time Reeds-Shepp motion primitives with continuous control primitives to allow for smoother trajectories during execution.

REFERENCES

[1] D. C. Conner, H. Kress-Gazit, H. Choset, A. A. Rizzi, and G. J. Pappas, "Valet parking without a valet," in *2007 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2007, pp. 572–577.

[2] K.-W. Min and J.-D. Choi, "Design and implementation of autonomous vehicle valet parking system," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 2082–2087.

[3] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[4] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5021–5028.

[5] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4889–4895.

[6] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, "Efficient sampling-based motion planning for on-road autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1961–1976, 2015.

[7] F. Islam, V. Narayanan, and M. Likhachev, "A*-connect: Bounded suboptimal bidirectional heuristic search," in *2016 IEEE International Conference On Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2752–2758.

[8] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[9] J. Chen, W. Zhan, and M. Tomizuka, "Constrained iterative lqr for on-road autonomous driving motion planning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–7.

[10] M. F. Hsieh and U. Ozguner, "A parking algorithm for an autonomous vehicle," in *2008 IEEE Intelligent Vehicles Symposium*. IEEE, 2008, pp. 1155–1160.

[11] R. Kummerle, D. Hahnel, D. Dolgov, S. Thrun, and W. Burgard, "Autonomous driving in a multi-level parking structure," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3395–3400.

[12] L. Han, Q. H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on rrt," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5622–5627.

[13] Q. Huy, S. Mita, and K. Yoneda, "A practical and optimal path planning for autonomous parking using fast marching algorithm and support vector machine," *IEICE TRANSACTIONS on Information and Systems*, vol. 96, no. 12, pp. 2795–2804, 2013.

[14] S. Klemm, M. Essinger, J. Oberländer, M. R. Zofka, F. Kuhnt, M. Weber, R. Kohlhaas, A. Kohs, A. Roennau, T. Schamm *et al.*, "Autonomous multi-story navigation for valet parking," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1126–1133.

[15] Y. Tazaki, H. Okuda, and T. Suzuki, "Parking trajectory planning using multiresolution state roadmaps," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 4, pp. 298–307, 2017.

[16] Y. Wang, "Improved a-search guided tree construction for kinodynamic planning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5530–5536.

[17] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[18] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-93-20, 1993.

[19] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An anytime, replanning algorithm," in *Proc. 2005 ICAPS*, 2005, pp. 262–271.

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

[21] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[22] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[23] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[24] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[25] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying mpc," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1586–1595, 2016.

[26] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332.

[27] J. Baltes, J. Bagot, S. Sadeghnejad, J. Anderson, and C.-H. Hsu, "Full-body motion planning for humanoid robots using rapidly exploring random trees," *KI-Künstliche Intelligenz*, vol. 30, no. 3-4, pp. 245–255, 2016.

[28] E. Páll, A. Sieverling, and O. Brock, "Contingent contact-based motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6615–6621.

[29] N. Chavan-Dafle and A. Rodriguez, "Sampling-based planning of in-hand manipulation with external pushes," in *Robotics Research*. Springer, 2020, pp. 523–539.

[30] M. Zucker, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "An optimization approach to rough terrain locomotion," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3589–3595.

[31] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2061–2067.

[32] S. Dai, M. Orton, S. Schaffert, A. Hofmann, and B. Williams, "Improving trajectory optimization using a roadmap framework," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 8674–8681.

[33] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, B. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[34] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[35] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[36] C. Chen, M. Rickert, and A. Knoll, "Kinodynamic motion planning with space-time exploration guided heuristic search for car-like robots in dynamic environments," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2666–2671.

[37] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn, "Search-based optimal motion planning for automated driving," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4523–4530.

[38] C. Chen, M. Rickert, and A. Knoll, "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 1148–1153.

[39] S. Klaudt, A. Zlocki, and L. Eckstein, "A-priori map information and path planning for automated valet-parking," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1770–1775.

[40] S. Koenig and M. Likhachev, "D^* lite," *AAAI*, vol. 15, 2002.

[41] X. Sun, W. Yeoh, and S. Koenig, "Moving target d* lite," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 2010, pp. 67–74.

[42] T. Oral and F. Polat, "Mod* lite: an incremental path planning algorithm taking care of multiple objectives," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 245–257, 2015.

[43] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1243–1248.

[44] B. Chandler and M. A. Goodrich, "Online rrt and online fmt: Rapid replanning with dynamic cost," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6313–6318.

[45] O. Adiyatov and H. A. Varol, "A novel rrt*-based algorithm for motion planning in dynamic environments," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2017, pp. 1416–1421.

[46] J. Qi, H. Yang, and H. Sun, "Mod-rrt*: A sampling-based algorithm for robot path planning in dynamic environment," *IEEE Transactions on Industrial Electronics*, 2020.

[47] W. Ding, W. Gao, K. Wang, and S. Shen, "Trajectory replanning for quadrotors using kinodynamic search and elastic optimization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7595–7602.

[48] ——, "An efficient b-spline-based kinodynamic replanning framework for quadrotors," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1287–1306, 2019.

[49] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.