Kinodynamic Motion Planning with Space-Time Exploration Guided Heuristic Search for Car-Like Robots in Dynamic Environments

Chao Chen[1] and Markus Rickert[1] and Alois Knoll[2]

***Abstract*— The Space Exploration Guided Heuristic Search (SEHS) method solves the motion planning problem, especially for car-like robots, in two steps: a circle-based space exploration in the workspace followed by a circle-guided heuristic search in the configuration space. This paper extends this approach for kinodynamic planning in dynamic environments by performing the exploration in both space and time domains. Thus, a time-dependent heuristic is constructed to guide the search algorithm applying a kinodynamic vehicle model. Furthermore, the search step-size and state resolution are adapted incrementally to guarantee resolution completeness with a trade-off for efficiency. The performance of Space-Time Exploration Guided Heuristic Search (STEHS) approach is verified in two scenarios and compared with several search-based and sampling-based methods.**

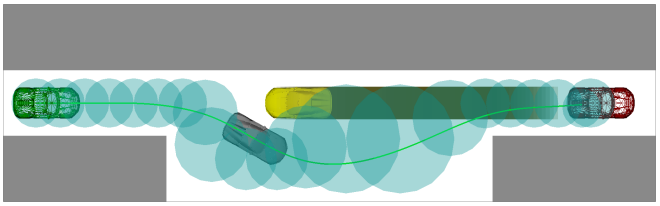

Fig. 1. Kinodynamic motion planning for a car-like robot with Space-Time Exploration Guided Heuristic Search (STEHS): The *green* and *red* vehicle frames represent the start and goal poses respectively. The static obstacles are colored in *gray* and the *yellow* vehicle is a dynamic object. The result of the space-time exploration is projected to the workspace as the *cyan* circles. The result motion is plotted in *green* lines. A snapshot is taken when the *gray* robot vehicle is performing the motion with the motion of the obstacle vehicle being demonstrated with the *yellow* shadows.

I. Introduction

Planning a feasible motion for a mobile-robot with non-holonomic constraints is a challenging task, especially in a dynamic environment [1]. If the time variation of the obstacles is unpredictable, a prompt replanning is required. Therefore, a real-time capable motion planner is important for a mobile-robot application with insufficient knowledge or imperfect sensing in dynamic environments. When information about the obstacle motion is available through object tracking or multi-agent communication, a kinodynamic motion planner is capable to plan a valid motion regarding the environment changes, as it considers the speed and acceleration of the robot to enable time-dependent collision checks with moving obstacles. The planner can also achieve better performance with the knowledge of dynamic environments, e.g., to help creating a heuristic for a search algorithm.

Fig. 1 shows a dynamic scenario in which a robot vehicle drives to a side track to let an approaching vehicle pass. The Space-Time Exploration Guided Heuristic Search (STEHS) planner explores the free-space regarding the moving obstacle and returns a path corridor connecting the start and goal positions. The heuristic search takes this result as a guidance to propagate the states with forward dynamics towards the goal configuration. There are several advantages of the STEHS approach:

- **Exploration-Based Space Decomposition**: The space-time exploration does not decompose the whole subspace as the combinatorial methods, but applies an A* algorithm to collect the free-space dimension and topology information in an appropriate scope. This knowledge is presented as a time-dependent heuristic, which balances path length and safety distance.
- **Kinodynamic Planning with Motion Primitives**: The heuristic search constructs a tree of states with forwards dynamics, which employs a kinodynamic model to propagate the states regarding a set of motion primitives. As a result, the solution can be directly executed as the control inputs for each motion step are provided.
- **Search Step-Size and Resolution Adaptation**: Regarding the free-space dimension along the path corridor, the search algorithm adapts the step-size and resolution. Furthermore, they are reduced incrementally when no solution is found, which makes a good trade-off between efficiency and completeness.
- **Grid-Free Heuristic Search**: The path corridor is presented in a decomposed form, which provides an easy clustering of the states. A list data structure maintains the states instead of a grid for the whole configuration space, which reduces the memory consumption.
- **Anytime Planning and Incremental Replanning**: As the exploration and search procedures are both performed in an A* search manner, the anytime or incremental techniques of the heuristic search can be applied. For example, an incremental exploration can be performed based on the previous result in replanning. The heuristic search can refine the step-size and resolution to improve the solution quality.

II. Related Work

In the context of kinodynamic planning for a car-like robot, not only the minimum turning radius constraint, but also the relations between velocity and acceleration, steering speed and orientation need to be considered. As

[1]Chao Chen and Markus Rickert are with fortiss GmbH, An-Institut Technische Universität München, Munich, Germany

[2]Alois Knoll is with Robotics and Embedded Systems, Technische Universität München, Munich, Germany

a result, there is no closed-form solution for the shortest path between two arbitrary configurations. The methods requiring explicit geodesics of the configuration space are impracticable, e.g., PRM [2]. The artificial potential field method [3] or the velocity obstacle model [4] are able to provide instant safe motions in dynamic environments by generating control inputs for local collision avoidance. However, a robot could get trapped in local minima for global motion. RRT provides a general approach for the kinodynamic planning with nonholonomic constraints by gradually constructing a search tree [5] [6]. However, the randomized exploration schema not only produces stochastic results with different qualities, but also causes performance issues in sophisticated environments, e.g., narrow passages. Among them RRT* [7] can achieve asymptotic optimality only when explicit connections between vertices are provided. The grid-based search methods [8] [9] solve the problem in a more systematic way by bearing a trade-off between completeness and performance in order to fit real-time applications. Moving obstacles can be modeled in a time-bounded lattice [10]. Furthermore, cooperation between different methods is developed for complex tasks, e.g., in [11] PRM and AD* search are combined for an anytime planning strategy in dynamic environments.

Additional knowledge about the configuration space or workspace can provide a great performance boost to the planning methods mentioned above. A global navigation function [12] or an elastic band [13] in the configuration space helps the potential field methods to avoid local minima. The workspace decomposition with wavefront expansion in [14] reduces the complexity of generating such a global guidance for the whole configuration space. Random sampling methods can optimize the sampling progress by collecting and evaluating the information about the configuration space with a low-dimensional grid projection or a subspace partition, e.g., EST [15], PDST [16] and KPIECE [17]. In [18], the balance between exploration and exploitation is emphasized and a workspace exploration is proposed to improve the exploration efficiency. The impact of workspace decompositions on randomized motion planning methods is studied in [19]. A grid discretization is typical space decomposition, which also provides a grid-based distance heuristic. However, further information such as free-space topology and the distance to obstacles is either implicit or not included. Moreover, a time-consuming pre-process is required to construct the grid, especially for a dynamic environment.

Applying the same principle of space guidance, the Space Exploration Guided Heuristic Search (SEHS) method [20] introduces a general space exploration procedure by expanding circles in a wave-like fashion from the start position to the goal position in the workspace. Thus, the planner gathers space topology knowledge of relevant areas, rather than the whole workspace. Meanwhile, the circle size indicates the local free-space dimension, which is useful for step-size adaptation in the search procedure. By clustering the states according to the circles, redundant states can be quickly identified regarding an adapted resolution. The generic SEHS approach is able to benefit from further traffic information, and environment changes can be handled by incremental replanning [21]. However, completeness is compromised by SEHS with a constant ratio between circle radius and step-size. Furthermore, time is not considered in the space exploration, as the workspace circles are created in a static environment. The Space-Time Exploration Guided Heuristic Search (STEHS) approach extends SEHS with an exploration including time domain and improves the algorithms in several aspects as presented in the following section.

III. General Space Exploration Guided Heuristic Search Framework

The Space Exploration Guided Heuristic Search framework consists of two steps: space exploration and heuristic search. This section introduces several further improvements such as A*-based space exploration and incremental adaptation of search resolution.

A. Space Exploration

The space exploration in a general SEHS framework investigates the free-space with simple geometric shapes for a passage from the start position to the goal position. The SEHS method takes circles for the atomic shapes as it relaxes the nonholonomic constraints of a car-like robot to a holonomic point robot with a certain safety margin. Algorithm 1 describes the exploration procedure in details.

A symbol c stands for a circle. A start circle c_{start} and a goal circle c_{goal} are given as initial conditions. The major modification to the algorithm in [20] is that a heuristic search is performed instead of the combined depth-first and breadth-first approach. The heuristic cost h of a circle is the Euclidean distance to the goal circle regarding the center points. The actual cost g is the accumulating distance from the start point through the centers of the predecessors. The total cost f is the sum of the both. Obviously, this heuristic is admissible and monotonic. Therefore, an open-set S_{open} and a closed-set S_{closed} can be applied. S_{open} holds the fresh circles sorted after the f-value. A function $\text{PopTop}(S_{\text{open}})$ picks the one with the minimum f-value from S_{open} as c_{current}. S_{closed} holds all the evaluated circles. A function $\text{Exist}(c_{\text{current}}, S_{\text{closed}})$ checks whether the center point of c_{current} is inside any circle from S_{closed}, except for its parent. If so, c_{current} is redundant because it can be covered by a circle from the closed-set and its descendants. Otherwise, a function $\text{Expand}(c_{\text{current}})$ creates child circles with centers on the border of c_{current}. In addition, a function $\text{Overlap}(c_{\text{current}}, c_{\text{goal}})$ checks if the current circle overlaps with the goal circle. The g-value and parent of the goal circle are updated if a shorter path is found. The algorithm terminates if the f-value of c_{current} is larger than the f-value of the goal circle. In this case an optimal solution is found.

B. Heuristic Search

The heuristic search takes the result from space exploration as a guidance, which provides a good compromise

Algorithm 1: SpaceTimeExploration($c_{\text{start}}, c_{\text{goal}}$)

$S_{\text{closed}} \leftarrow \emptyset$;
$S_{\text{open}} \leftarrow \{c_{\text{start}}\}$;
while $S_{\text{open}} \neq \emptyset$ **do**
  $c_{\text{current}} \leftarrow \text{PopTop}(S_{\text{open}})$;
  if $f[c_{\text{goal}}] < f[c_{\text{current}}]$ **then**
    return success;
  else if !Exist($c_{\text{current}}, S_{\text{closed}}$) **then**
    $S_{\text{open}} \leftarrow \text{Expand}(c_{\text{current}}) \cup S_{\text{open}}$;
    if Overlap($c_{\text{current}}, c_{\text{goal}}$) **then**
      if $f[c_{\text{current}}] < g[c_{\text{goal}}]$ **then**
        $g[c_{\text{goal}}] = f[c_{\text{current}}]$;
        $\text{parent}[c_{\text{goal}}] = c_{\text{current}}$;
    $S_{\text{closed}} \leftarrow \{c_{\text{current}}\} \cup S_{\text{closed}}$;
return failure;

Algorithm 2: HeuristicSearch($\{c_i\}, \vec{q}_{\text{start}}, \vec{q}_{\text{goal}}$)

$S_{\text{closed}} \leftarrow \emptyset$;
$S_{\text{open}} \leftarrow \{\vec{q}_{start}\}$;
$k \leftarrow k_{\text{init}}$;
while $S_{\text{open}} \neq \emptyset$ **do**
  $\vec{q}_{\text{current}} \leftarrow \text{PopTop}(S_{\text{open}})$;
  $c_{\text{current}} \leftarrow \text{MapNearest}(\vec{q}_{\text{current}})$;
  if $f[\vec{q}_{\text{goal}}] < f[\vec{q}_{\text{current}}]$ **then**
    return success;
  else if !Exist($\vec{q}_{\text{current}}, c_{\text{current}}, k$) **then**
    $S_{\text{open}} \leftarrow \text{Expand}(\vec{q}_{\text{current}}, c_{\text{current}}, k) \cup S_{\text{open}}$;
    if $h[\vec{q}_{\text{current}}] < R_{\text{goal}}$ **then**
      GoalExpand($\vec{q}_{\text{current}}, \vec{q}_{\text{goal}}$);
    $S_{\text{closed}} \leftarrow \{\vec{q}_{\text{current}}\} \cup S_{\text{closed}}$;
  if $S_{\text{open}} = \emptyset$ **then**
    $k = k \times 0.5$;
    if $k > k_{\min}$ **then**
      $S_{\text{open}} \leftarrow S_{\text{closed}}$;
      $S_{\text{closed}} \leftarrow \emptyset$;
return failure;

between path length and safety distance. Furthermore, the step-size and resolution are incrementally adapted to achieve a better balance between performance and completeness. The heuristic search is defined as Algorithm 2.

The heuristic search with open-set and closed-set is similar as Algorithm 1. In each iteration, the state with the smallest f-value from the open-set is evaluated. A function MapNearest($\vec{q}_{\text{current}}$) maps the selected state $\vec{q}_{\text{current}}$ to the nearest circle. The h-value is the sum of the distance from $\vec{q}_{\text{current}}$ to the next circle and the distance along the rest circle centers to the goal state. Thus, the next circle of the path corridor is chosen as a local target for the states propagation. If the mapped circle is the goal circle, the Euclidean distance to the goal state is taken as the h-value. A function Expand($\vec{q}_{\text{current}}, c_{\text{current}}, k$) applies predefined primitive motions and checks collisions to create new states. When $\vec{q}_{\text{current}}$ is inside a defined goal range R_{goal}, a function GoalExpand($\vec{q}_{\text{current}}, \vec{q}_{\text{goal}}$) tries to directly reach the goal with primitive motions. The redundancy are resolved by the function Exist($\vec{q}_{\text{current}}, c_{\text{current}}, k$), which evaluates all the states mapped to the same circle.

The circle radius is used to adapt the motion step-size and the state resolution. An improvement to SEHS in [20] is that a *step-rate* parameter k is introduced to enable an incremental refinement in function Expand($\vec{q}_{\text{current}}, c_{\text{current}}, k$), and Exist($\vec{q}_{\text{current}}, c_{\text{current}}, k$). Starting with an initial value k_{init}, k is halved if the open-set is empty. Then, the search restarts by adopting the closed-set as the new open-set. The algorithm terminates with no solution only when k reaches a lower bound $k_{\min}$. Thus, the heuristic search begins with a large step-size and resolution, and then gradually refines them when necessary, which achieves a good trade-off between completeness and efficiency.

IV. Cylinder-Based Space-Time Exploration

In dynamic environments, the velocity of a robot should also be taken into account. Starting from a single point, a point robot with a constant velocity v can reach an area of a circle with radius r in a time duration $\Delta t = r/v$. The time evolution of the reachable set can be modeled with a cone. As shown in Fig. 2, a cone grows from the point O_0 to a height of Δt and a radius of r. A child cone can start from any point inside the parent cone. For example, a new cone is created at the point O_1 on the top of the cone from O_0. If the timestamp at O_0 is t, the timestamp at O_1 is $t + \Delta t$. The cone from O_0 exists in the time-slot $[t, t + \Delta t]$.

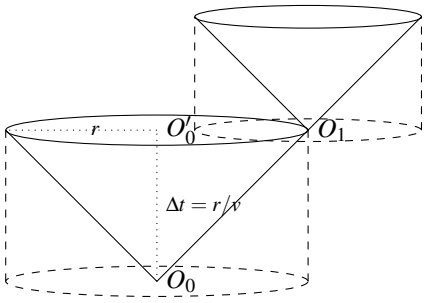

Fig. 2. Modeling of space and time using cylinder-based exploration. A cone represents the reachable set for a robot within a given time frame and is approximated by a cylinder in the dashed lines.

A cone grows until it reaches an obstacle with a safety margin. However, it is complex to perform this procedure with a time-dependent collision test or distance query for cones. A simpler way is to approximate the cones with cylinders, as the dashed lines in Fig. 2. The radius of the cylinder is decided in an iterative manner. First, a collision-free distance d_t is calculated at the starting time t. Then, the time duration is obtained with $\Delta t = d_t/v$. After

that, the distance to obstacles is checked again in time-slot $[t, t+\Delta t]$ with the sweeping area of the moving obstacles. $\min(d_t, d_{t+\Delta t})$ is the radius of the cylinder. If the distance $d_{t+\Delta t}$ is chosen, the time duration is reduced to $d_{t+\Delta t}/v$. As a cylinder is an overestimation of the total reachable set of the point robot in a cone, it contains all the possible solutions. Thus, the two-dimensional circles in SEHS are extended to three-dimensional cylinders for the space-time exploration.

In this case, the circles in Algorithm 1 are replaced with cylinders for space-time exploration, i.e., a symbol c stands for a cylinder. The costs are measured in time. The h-value of a cylinder is the projected Euclidean distance divided by a desired velocity for the robot motion. The heuristic is still admissible and monotonic as the distance is divided by one positive value for velocity. The g-value can be directly obtained as the difference between the cylinder timestamp and the start timestamp. The redundant cylinder is identified by examining whether the starting point of the cylinder is inside any cylinder from the closed-set, except for its parent. The child cylinders are generated on top of the parent cylinder, with starting points locating at the center point and the interpolated points on the top circle.

Fig. 3 shows the space-time exploration result of the scenario in Fig. 1. The expanded yellow cylinders approximate the total reachable set of the robot. The result is a cyan cylinder path in the reachable set. The space-time exploration can be carried out in an incremental manner, based on the space exploration result of the static environment. The path circles are first converted to space-time cylinders. If there is a gap between two neighboring cylinders, a local space-time exploration is performed to fix the connection. Thus, the space-time exploration is performed only when a moving obstacle blocking the path.

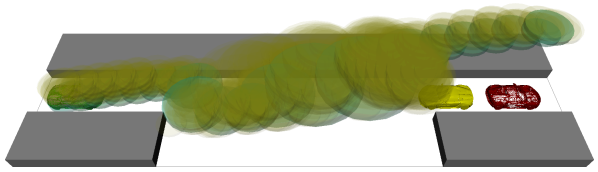

Fig. 3. Cylinder-based space-time exploration: The transparent *yellow* cylinders are expanded during the exploration. The transparent *cyan* cylinders show the resulting cylinder path.

V. Cylinder Path Guided Heuristic Search

The space-time heuristic search is similar as Algorithm 2, following the cylinder path with modified distance functions. The distance between a state and a cylinder is the combination of the space distance (1) and the time distance (2). The space distance is calculated with the projections of the state and the cylinder in the two-dimensional Cartesian space. The time distance is the time difference between the state timestamp and the cylinder time-slot. The total distance is calculated with (3). v is a desire velocity identical as in space-time exploration. The distance between the cylinders of the path corridor is the difference between their timestamps.

$$d_{\text{space}}(\vec{q}, c) = \max(\|(x_{\vec{q}}, y_{\vec{q}}) - (x_c, y_c)\| - r_c, 0)\,. \tag{1}$$

$$d_{\text{time}}(\vec{q}, c) = \begin{cases} t_c - t_{\vec{q}} & \text{if } t_{\vec{q}} < t_c \\ t_{\vec{q}} - (t_c + \Delta t_c) & \text{if } t_{\vec{q}} > (t_c + \Delta t_c) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$d_{\text{total}}(\vec{q}, c) = \frac{d_{\text{space}}(\vec{q}, c)}{v} + d_{\text{time}}(\vec{q}, c) \tag{3}$$

The mapping between a state and the nearest cylinder is based on the total distance (3). The distance and time difference between two states are resolved with (4) and (5) respectively. The distance (4) combines the projected point distance $\|(x_{\vec{q}_i}, y_{\vec{q}_i}) - (x_{\vec{q}_j}, y_{\vec{q}_j})\|$ and the orientation difference $|\theta_{\vec{q}_i} - \theta_{\vec{q}_j}|$. The orientation difference is multiplied with the minimum turning radius $r_{\min}$ to calculate the minimum traveling distance to compensate the angle discrepancy. The maximum operation returns a lower bound of the motion distance from one state to the other. The time difference is $|t_{\vec{q}_i} - t_{\vec{q}_j}|$. If the distances are smaller than a certain value, r_{distance} and r_{time} respectively, one state is redundant. The resolution values are proportional to the cylinder radius and height, as well as the step-size in Algorithm 2.

$$d_{\text{space}}(\vec{q}_i, \vec{q}_j) = \max(\|(x_{\vec{q}_i}, y_{\vec{q}_i}) - (x_{\vec{q}_j}, y_{\vec{q}_j})\|, |\theta_{\vec{q}_i} - \theta_{\vec{q}_j}| r_{\min}) \tag{4}$$

$$d_{\text{time}}(\vec{q}_i, \vec{q}_j) = |t_{\vec{q}_i} - t_{\vec{q}_j}| \tag{5}$$

Fig. 4 shows the heuristic search result based on the cylinder path from Section IV. A kinodynamic vehicle model is applied with a constant velocity and different change rates for curvature. We can see that the planner takes large steps in large cylinders, and the expanded states follow the path cylinders. The result robot motion pursues the centers of the cylinders to achieve a good safety distance.

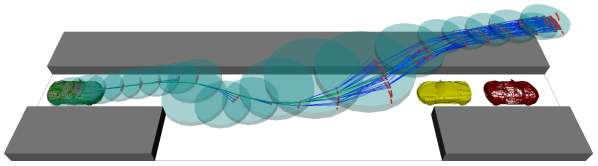

Fig. 4. Cylinder path guided heuristic search: The transparent *cyan* cylinders are the cylinder path from the space-time exploration. The *red* points are the expanded states connected with *blue* lines. The *green* lines show the solution from the search algorithm.

VI. Dynamic Environment Experiments

In this section, the STEHS planner is verified in two dynamic scenarios[1], comparing with the search-based SEHS and Hybrid A* methods, as well as the random-sampling RRT, EST, PDST, and KPIECE algorithms from the Open Motion Planning Library [22]. All the planners apply the

[1] https://www.youtube.com/watch?v=AmyweePd1HU

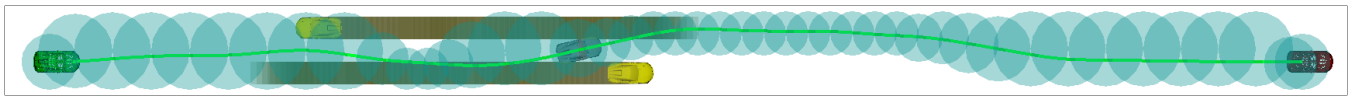

Fig. 5. Overtaking scenario: A robot vehicle (*gray*) overtaking a moving obstacle (yellow vehicle on bottom) in front, while another moving obstacle (*yellow* vehicle on top) is approaching from the opposite direction.

same implementation of a kinodynamic vehicle model introduced in [20] and the same environment model with collision check functions. An objects list holds the static and dynamic obstacles. The motion of the dynamic obstacles is known, which enables collision checks with any time-labeled state. The SEHS algorithm performs the space exploration only in the static environment. So does the Hybrid A* calculate the grid-based heuristic. The dynamic obstacles are evaluated later during the states expansion. The performance is measured with 100 trials with random start positions in an area of $2\,\mathrm{m} \times 2\,\mathrm{m}$. The experiments are conducted on a computer with an Intel Core i7 2.90 GHz processor and 8 GB RAM running Linux.

TABLE I

RESULTS OF THE OVERTAKING SCENARIO

Planner	States	Verifications	Time (ms)	STD (ms)
$\mathrm{STEHS}_{explore}$	961	1796	3.59	2.36
STEHS_{search}	545	2553	11.27	4.73
$\mathrm{SEHS}_{explore}$	627	1148	1.42	0.68
SEHS_{search}	1107	6613	24.92	30.29
Hybrid A*	4033	35162	180.16	283.14
RRT	1828	239488	414.18	261.84
EST	8618	1627933	2274.91	1645.96
PDST	978	62920	187.15	98.54
KPIECE	1822385	3717245	7164.50	4493.49

A. Overtaking Scenario

The first example is an overtaking scenario shown in Fig. 5. A robot vehicle and an obstacle vehicle are moving in the same direction along a road. As the ego vehicle moves faster than the obstacle, it has to perform an overtaking maneuver to reach the goal position. At the same time, another obstacle vehicle is approaching in the opposite direction on the other side of the road. Therefore, the planner should plan an overtaking motion without colliding with the approaching obstacle. The search-based methods apply a set of motion primitives with combinations of 3 different acceleration and 3 different steering values as in [20]. Fig. 5 is a snapshot of the robot vehicle performing the result motion from STEHS. The cyan circles are the projections of the path cylinders from the space-time exploration. The green line is the overtaken motion planned by heuristic search. The motion of the dynamic obstacle is demonstrated with the blurring trails.

Table I compares the simulation results from the different algorithms via the number of states, the number of collision tests, and the planning time with its standard deviation. The STEHS and SEHS methods have two rows for the space exploration and heuristic search respectively. The numbers in the exploration row are the counts of circles or cylinders and the distance queries. Regarding the planning time, the STEHS method takes half of the total planning time compared to the SEHS method on average. Although the space-time exploration costs more time than the space exploration in SEHS, it provides better heuristics for the dynamic scenario, that more time is saved in the search phase. The SEHS performs better than Hybrid A* in planning time, as the incremental adaptation of the search step and resolution reduce the number of nodes to compensate for the sub-optimal heuristic estimation. In contrast, the step size and resolution of Hybrid A* search are constant. Therefore, it needs a small resolution of 0.3 m in both x and y direction to solve all the random scenarios. These two search-based methods both have large deviations of planning time, because the dynamic obstacles create a narrow passage in a certain time-slot which is occasionally hard to traverse with the static heuristic. The STEHS method extracts this knowledge already in the exploration phase to adjust the search direction and the step-size fitting to the path corridor.

The solutions from the random-sampling methods vary greatly in planning duration and geometric form of the path. The robot position and orientation is chosen as the subspace for the space projections in EST, PDST, and KPIECE. The subspace projection and the progress evaluation do not always work well in dynamic scenarios. Only PDST has better time performance than RRT. As the subspace occupation is dynamic in this scenario, the workspace projection does not always provide the right information for a certain moment. The KPIECE method often suffers from the numerical precision limitation problem and only returns an approximate solution. Furthermore, these random-sampling methods cannot provide any optimal guarantee. The RRT* and PRM* methods are not applicable to the kinodynamic motion model with nonholonomic constraints—there is no closed-form solution to connect two specific states in the configuration space to optimize the path length.

B. Intersection Scenario

The second example takes place in an intersection as shown in Fig. 6. The robot vehicle needs to move across the intersection to reach the goal position, while two obstacle vehicles are coming through the intersection at the same time. The speed of the vehicles is configured to cause a collision in the middle of the intersection, if the ego vehicle does not adapt its motion. The planner should plan a motion to avoid the collision and reach the goal. A snapshot of the result motion is shown in Fig. 6. The quantitative results from the planners are listed in Table II.

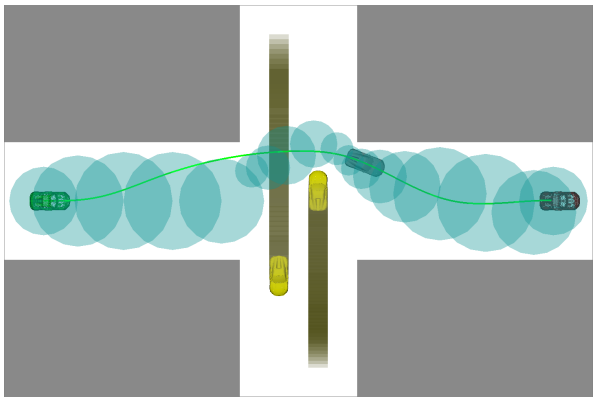

Fig. 6. Intersection scenario: A robot vehicle (*gray*) moves across an intersection to reach the goal, while two obstacles vehicles (*yellow*) are crossing the intersection at the same time.

TABLE II

RESULTS OF THE INTERSECTION SCENARIO

Planner	States	Verifications	Time (ms)	STD (ms)
STEHS$_{explore}$	1070	1571	4.50	2.39
STEHS$_{search}$	2666	17265	67.28	44.54
SEHS$_{explore}$	405	520	0.64	0.57
SEHS$_{search}$	2825	27064	101.72	122.53
Hybrid A*	18234	181418	1311.32	1664.42
RRT	2591	337998	785.19	921.81
EST	4219	843493	1444.03	2087.31
PDST	2212	149052	453.22	366.32
KPIECE	1820488	2883336	8001.44	3648.32

The ranking of the results is the same as the first scenario. The advantage of STEHS is reduced comparing to the first scenario with a larger variation in planning time. This effect is due to the fact that the space-time exploration does not consider the vehicle kinematics. The path corridor captures the obstacle dynamic, but may be sub-optimal for the vehicle to follow due to the nonholonomic and differential constraints. This effect is more obvious in the intersection scenario, where a dramatic evasion is required in a small free-space.

VII. CONCLUSION

The Space-Time Exploration Guided Heuristic Search approach combines time-dependent workspace exploration with configuration space heuristic search to solve the motion planning problem for car-like robots in dynamic environments. This method improves the SEHS approach with time modeling in space exploration and a step-rate factor for incremental search step-size and resolution adaptations. The space-time exploration creates a cylinder path regarding the desired velocity of the robot and the dynamics of the environment. The search procedure follows this time-dependent heuristic to expand states for a solution. The cylinder path from the exploration phase provides a good compromise between path length and safety distance, while the step-rate adaptation achieves a fine balance between time-performance and resolution completeness.

As future work, the STEHS planner will be evaluated for incremental replanning with environment updates, and imperfect perception with uncertainties.

REFERENCES

[1] J. Canny, "Complexity of robot motion planning," Ph.D. dissertation, Massachusetts Institute of Technology, 1987.

[2] L. Kavraki, P. Švestka, J.-C. Latombe, and M. 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.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[4] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[5] S. LaValle and J. Kuffner Jr., "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[6] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[7] 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.

[8] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[9] 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.

[10] A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," in *Proc. IEEE International Conference on Robotics and Automation*, 2009, pp. 1662–1668.

[11] Jur van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proc. IEEE International Conference on Robotics and Automation*, 2006, pp. 2366–2371.

[12] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–648, 1991.

[13] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. IEEE International Conference on Robotics and Automation*, 1993, pp. 802–807.

[14] O. Brock and L. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *Proc. IEEE International Conference on Robotics and Automation*, 2001, pp. 1469–1474.

[15] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *International Journal of Computational Geometry and Applications*, vol. 9, no. 4-5, p. 495512, 1999.

[16] A. Ladd and L. Kavraki, "Motion planning in the presence of drift, underactuation and discrete system changes," in *Proc. Robotics: Science and Systems*, 2005, pp. 233–241.

[17] I. Şucan and L. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Workshop on the Algorithmic Foundations of Robotics*, 2008.

[18] M. Rickert, O. Brock, and A. Knoll, "Balancing exploration and exploitation in motion planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2008, pp. 2812–2817.

[19] E. Plaku, L. Kavraki, and M. Vardi, "Impact of workspace decompositions on discrete search leading continuous exploration (dslx) motion planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2008, pp. 3751–3756.

[20] C. Chen, M. Rickert, and A. Knoll, "Combining space exploration and heuristic search in online motion planning for nonholonomic vehicles," in *Proc. IEEE Intelligent Vehicles Symposium*, 2013, pp. 1307–1312.

[21] ——, "A traffic knowledge aided vehicle motion planning engine based on space exploration guided heuristic search," in *Proc. IEEE Intelligent Vehicles Symposium*, 2014, pp. 535–540.

[22] I. Şucan, M. Moll, and L. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012, http://ompl.kavrakilab.org.