Combining Space Exploration and Heuristic Search in Online Motion Planning for Nonholonomic Vehicles

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

***Abstract*— This paper presents an efficient motion planning method for nonholonomic vehicles, which combines space exploration and heuristic search to achieve online performance. The space exploration employs simple geometric shapes to investigate the collision-free space for the dimension and topology information. Then, the heuristic search is guided by this knowledge to generate vehicle motions under kinodynamic constraints. The overall performance of this framework greatly benefits from the cooperation of these two simple generic algorithms in suitable domains, which sequentially handles the free-space information and kinodynamic constraints. Experimental results show that this method is able to generate motions for nonholonomic vehicles in a time frame of less than 100 milliseconds for the given problem settings.**

The contribution of this work is the development of a Space Exploration Guided Heuristic Search with a circle-path based heuristics and adaptable search step size. The approach is grid-free and able to plan nonholonomic vehicle motions under kinodynamic constraints.

I. Introduction

Due to the wide applications of mobile platforms and autonomous vehicles, motion planning (combining path planning and trajectory generation) for car-like robots or nonholonomic vehicles always stays in the focus of robotics and intelligent vehicles research and development.

In general, two kinds of information should be processed during the motion planning: constraints from the environment and constraints from the vehicle itself. The former one determines the collision-free subspace $\mathscr{C}_{\text{free}}$ of the vehicle configuration space $\mathscr{C}$. The latter one defines the dimension and metric of $\mathscr{C}$. The mission of motion planning is to find a collision-free path $\{q_0, q_1, \ldots, q_n\}$ from start configuration q_{start} to goal q_{goal} together with a series of time-labeled control variables $\{u_{t_0}, u_{t_1}, \ldots, u_{t_1}\}$, which could drive the vehicle through this path.

In classic motion planning paradigms, a collision-free path is planned in advance only considering the kinematic constraints. After that, a trajectory is generated according to the speed and acceleration constraints. However, these methods do not always produce optimal results. Because the path generated in the first place may be suboptimal under certain speed and turning conditions or acceleration and steering constraints of the vehicle. Even worse, the path may contain inevitable collisions, which can only be found by trajectory generation and require modifications of the original path. A kinodynamic planner, which also takes speed and steering into account, performs better in such situations. As the price, a kinodynamic planner faces a more complicated search space containing speed and steering, which makes it more challenging to achieve online performance. The related work is introduced in Section II.

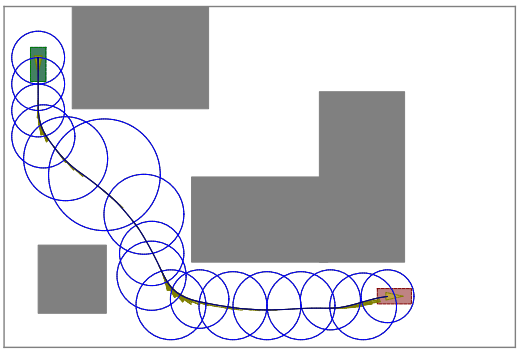


Fig. 1. Motion planning for nonholonomic vehicles using Space Exploration Guided Heuristic Search. Start position is shown in *green*, goal in *red*, obstacles in *gray*, exploration circles and solution trajectory in *blue*.

This paper introduces a novel online capable kinodynamic planner. Compared to the traditional approaches, this method relaxes the kinematic constraints in the first path-finding stage in order to quickly extract free-space knowledge; it then employs a heuristic search algorithm to find a trajectory through the free-space. The kinodynamic constraints are handled during the search node expansion with a numerical integration-based forward dynamic method. The combination between these two steps is realized by a circle-path based heuristic design and search step size adaptation. Thus, the search space is customized to the $\mathscr{C}_{\text{free}}$, which greatly improves the search efficiency even without $\mathscr{C}$ discretization. The resulting motion is applied to the kinodynamic constraints and optimal for safety regarding the distance to obstacles. The details of the whole framework are elaborated in Section III.

An example is demonstrated in Fig. 1. The circles are generated by the space exploration, which indicates a "circle path" in free-space. The curved lines are expanded during the heuristic search, which drives from start to goal through the circle-path. Further experiments and comparison with reference methods are shown in Section IV

II. Related Work

The idea of space exploration has played an important role in many motion planning approaches. As planning is always done in $\mathscr{C}$, the exploration is first targeted in $\mathscr{C}$

[1]Chao Chen and Markus Rickert are with fortiss GmbH, affiliated institute of Technische Universität München, Munich, Germany

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

to extract the information of $\mathscr{C}_{\text{free}}$. As the kinematics of a car-like robot is a small-time controllable system [1], it is possible to design a steering method, which drives the vehicle through any collision-free corridor in $\mathscr{C}_{\text{free}}$. This collision-free corridor can be easily obtained by a geometric planner in $\mathscr{C}$. However, the kinodynamic constraints make a car-like robot only locally controllable, with the consequence that a geometric path does not guarantee an applicable trajectory. Potential fields, such as the artificial potential field in [2], can handle the kinodynamic constraints well with real-time performance and superposition ability. Because of local minima, such methods are limited to local problems, otherwise the whole $\mathscr{C}_{\text{free}}$ should be analyzed for a global navigation function [3]. The overhead of such analysis is too large for online applications. Another approach is to partially evaluate the $\mathscr{C}_{\text{free}}$ to achieve guidance for potential fields, such as elastic bands in [4]. But as long as the exploration is done in $\mathscr{C}$, the complexity grows exponentially with the space dimension. In [5], Brock and Kavraki suggested to explore the workspace $\mathscr{W}$ instead. The $\mathscr{W}$ of vehicle motion planning is the 3D Cartesian space, and even 2D is sufficient in most scenarios. As a result, the exploration turns out to be much more efficient, and more complicated structures can be built for the planning [6].

The metric in $\mathscr{C}$ of a nonholonomic vehicle is non-trivial, which leaves a large gap between learning the geometric path in $\mathscr{C}_{\text{free}}$ and finding a drivable path under the constraints. The works from Dubins [7] and Reeds & Shepp [8] provide an insight into these complex metrics. With the assumption that the vehicle can only steer with a certain radius, the metric can be described with a bunch of combinations of primitive segments such as straight lines and arcs. These are however suboptimal in practice, because the vehicle must stop and steer at every connecting point of two segments. These is no explicit method to obtain metric between two configurations when kinodynamic constraints are considered. A workaround is provided through forward kinematics or dynamics, which provide neighbor configurations according to certain control inputs. With this technique, kinodynamic planning is possible, for example, the Rapidly-Exploring Random Tree (RRT) [9] and its variants. The exploration efficiency is a key factor for RRTs, therefore numerous exploration schemes have been developed. RRT-Connect [10] takes two trees to double the exploration performance. RRT-Blossom [11] uses a flood-like method with multiple expansions. Shkolnik et al. [12] use reachability information to guide the exploration. In contrast, the information obtained from the continuous exploration could also update the discrete lead decisions [13]. Workspace exploration is applied by Rickert et al. [14], which is especially powerful to solve the narrow passage problems. Furthermore, RRT* [15] introduced an any-time algorithm which optimizes the result continuously during the runtime. However, due to the randomness of the planning results, these sample-based methods are less preferable for practical autonomous driving applications. And time requirements exceed the online criterion in most cases.

In recent years, grid-based search methods are customized for motion planning of intelligent vehicles, e.g., any-time incremental search in multi-resolution grid [16] and Hybrid A* search [17]. These methods made trade-off between completeness and performance by discretizing $\mathscr{C}$ and applying heuristic search with predefined primitive motions to expand the vehicle states. In addition, other efforts have been made in trajectory generation when a path is provided [18], which focuses on the kinodynamic constraints and human-comfort. The heuristics and cost function design is the crucial part of these methods. For nonholonomic and kinodynamic systems, finding a good heuristic is almost as complex as the motion planning problem itself. Furthermore, the grid in $\mathscr{C}$ is anisotropic due to the complicated metric structure and different resolutions are required for different situations. However, theses methods show great advantages in online performance, because the discretization and the heuristics greatly reduce the search space.

The idea of this paper is to combine an exploration stage in $\mathscr{W}$ with a grid-free heuristic search in $\mathscr{C}_{\text{free}}$. The space exploration can rapidly check the existence of possible solution by examining the space connectivities. By constructing the space connection topology, not only a distance heuristic is available for the search phase, but also the information of the free-space dimension can be useful for choosing the optimal search step size. This method can be regarded as the search-based version of [5] and [14]. Details are explained in the following sections.

III. Exploration and Search Framework

A. Problem Statement

The kinodynamic state of a vehicle is (x, y, θ, v, ϕ), which can also be expressed as a vector q in $\mathscr{C}$. The vehicle takes (a, ω) as control inputs, i.e., control vector u. The speed v and steering angle ϕ are bounded, as is the acceleration a and steering speed ω. According to the bicycle model, the kinodynamic motion is specified as

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} v\cos\theta\cos\phi \\ v\sin\theta\cos\phi \\ v\sin\phi/l \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} a + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \omega . \quad (1)$$

The first column on the right is the drift of the system, which means the vehicle will continue changing its state even if the control inputs are equal to zero. l is a constant value of the vehicle axis-base. The turning radius r is determined by l and ϕ with the equation $r = l\cot\phi$.

The problem statement of motion planning is as follows: given a start configuration q_{start} and a goal configuration q_{goal} in $\mathscr{C}$, the desired planning result is a series of control inputs $\{u_t\}$, which drives the vehicle from q_{start} to q_{goal}. All the intermediate states $\{q_t\}$ should be in $\mathscr{C}_{\text{free}}$, which consist the trajectory of the motion.

If no sliding occurs during the motion, the track from a high-speed motion can always be followed by a motion with lower speed. A short explanation is as follows: when

traveling with bounded steering speed and acceleration, the path from the vehicle is smooth and determined by the curvature. The curvature of the path is continuous. When traveling slower through the same path, the time derivative of the curvature becomes smaller, which results in slower changes of ϕ, i.e., smaller ω. Thus, the problem condition can be simplified as finding a trajectory for a q^*_{goal} with $v^*_{\text{goal}} \geq v_{\text{goal}}$.

B. Space Exploration

The space exploration is done in $\mathscr{W}$, i.e., the 2D Cartesian space. The basic geometric shape used in the exploration is circles, which have the simplest collision detection and distance calculation. The circles are expanded from the start to goal position with a depth-first schema taking the Euclidean distance as guidance. Meanwhile, the largest circle is also visited in each iteration, which serves as a breadth-first exploration to maximize the space explored, as well as to escape local minima. New circles are born on the edge of the parent circle and take the radius as the distance to the nearest obstacle. A detailed explanation is given in [19].

Partial collision detections are done with a distance margin as the inner circle radius of the vehicle geometry. The space exploration procedure is shown in Algorithm 1.

Algorithm 1 SpaceExploration$((x,y)_{\text{start}}, (x,y)_{\text{goal}})$

```
c_start ← GenerateCircle((x,y)_start)
c_goal ← GenerateCircle((x,y)_goal)
S_closed ← ∅
S_open ← c_start
while S_open ≠ ∅ do
    if Overlap(Nearest(S_open), c_goal) then
        return success
    else
        c_nearest ← PopNearest(S_open)
        c_largest ← PopLargest(S_open)
        if NotExist(c_nearest) then
            S_open ← Expand(c_nearest)
            S_closed ← c_nearest
        end if
        if NotExist(c_largest) then
            S_open ← Expand(c_largest)
            S_closed ← c_largest
        end if
    end if
end while
return failed
```

Two sets of circles are maintained by the algorithm: S_{closed} holds all the visited circles for overlapping check or result construction; S_{Open} contains all the newborn circles for expansion. The function GenerateCircle$((x,y))$ creates a free-space circle centered at the position (x,y) with the radius equal to the distance to the nearest obstacle minus the inner radius of the vehicle. The function Overlap(c_1,c_2) checks if two circles overlap with each other in a certain margin, e.g., 50% of the radius of the smaller circle, which guarantees enough space for a transition motion. The function PopNearest(S) and PopLargest(S) pop the circle nearest to goal or with the largest radius respectively from set S. The function NotExist(c) checks whether the circle c greatly overlaps with any circle in S_{closed}. In this case, the circle is neglected in order to reduce the redundancy. The function Expand(c) samples the border of circle c, and generates child-circles on the sample points with the function GenerateCircle$((x,y))$. When c_{goal} is reached, the program traces back the relationship among circles to produce and optimize a circle-path. A path distance is also calculated for each circle according to the distance between circle centres.

The circles are grown in a tree-like manner. No other structures among circles are built based on intersections. The interconnections are only examined in the final result optimization. A more complicated exploration scheme could be building a graph data structure and find the shortest path with Dijkstra's algorithm, or growing the circles from both start and goal sides. The experiment section will show that even without these enhancements, the simple version of space exploration is fast and efficient.

C. Heuristic Search

The heuristic search makes use of the circle-path from space exploration. This rough path of overlapping circles provides two useful informations: distance estimations and hints of free-space dimension.

The distance is used to build a heuristic of estimated time cost from q to q_{goal} with

$$t_{\text{heuristic}} = \frac{d_{q,c} + d_c}{|v|}. \tag{2}$$

$d_{q,c}$ is the distance between configuration q and circle c. d_c is the circle-path distance from circle c to goal. The sum of both is the estimated distance from q to q_{goal}. It is then divided by the current vehicle speed v to obtain estimated time cost. For each q, multiple time estimations are calculated with the nearest circle and its neighbors, and the minimum is taken as the valid heuristic cost. Time is used instead of distance as a heuristic due to the fact that speed matters in kinodynamic planning. With this heuristic, the node with the faster speed has higher priority when distances are similar. The search algorithm based on this heuristic is described in Algorithm 2. As the typical heuristic algorithms, the node from the open set S_{open} with the smallest total cost is expanded in each iteration. The total time cost is calculated by adding the heuristic time cost with the time already spent to reach q together. The visited node is then pushed to the closed set S_{closed}, while the new ones are pushed to S_{open}. This procedure is repeated until S_{open} is empty or q_{goal} is reached.

The function Expand(q,s) generates new configurations from q with a step size s. The size of the circle helps to select the right s with

$$s = \min(\alpha \cdot r_{\text{c}}, \beta \cdot d_{\text{c}}, s_{\min}). \tag{3}$$

α and β are coefficients and r_{c} is the radius of the circle which associates with the heuristic value. d_{c} is the circle-path distance and $s_{\min}$ is a pre-defined minimal step size. By

Algorithm 2 HeuristicSearch$(q_{\text{start}}, q_{\text{goal}}, \{c_i\})$

$S_{\text{closed}} \leftarrow \emptyset$
$S_{\text{open}} \leftarrow q_{start}$
$i \leftarrow 0$
while $S_{\text{open}} \neq \emptyset$ **do**
  if $\text{Top}(S_{\text{open}}) \approx q_{\text{goal}}$ **then**
    return success
  else
    $q \leftarrow \text{PopTop}(S_{\text{open}})$
    if $\text{NotExist}(q)$ **then**
      $s \leftarrow \text{UpdateStep}(c_i, q_{\text{goal}}, q)$
      $S_{\text{open}} \leftarrow \text{Expand}(q, s)$
      $S_{\text{closed}} \leftarrow q$
    end if
  end if
end while
return failed

taking the minimum of the three, the planner tends to take large steps with big circles, which indicate vast free-space, and proceeds with small steps when it is close to the goal.

The child configurations are generated with forward dynamics as

$$\begin{pmatrix} x_{t+\Delta t} \\ y_{t+\Delta t} \\ \theta_{t+\Delta t} \\ v_{t+\Delta t} \\ \phi_{t+\Delta t} \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ \theta_t \\ v_t \\ \phi_t \end{pmatrix} + \begin{pmatrix} v\cos\theta\cos\phi \\ v\sin\theta\cos\phi \\ v\sin\phi/l \\ a \\ \omega \end{pmatrix} \Delta t\,. \quad (4)$$

Δt is the time step to perform the numeric integration. The number of integration steps are decided with equation $n \approx s/|v|/\Delta t$. The value of the control variables a and ω are chosen from a group of pre-defined primitive actions. Table I shows a simple example with combinations of three acceleration and three steering speed values.

TABLE I
ELEMENTAL ACTIONS FOR THE EXPANSION PHASE

	$a>0$	$a=0$	$a<0$
$\omega>0$	accelerate_left	left	decelerate_left
$\omega=0$	accelerate	straight	decelerate
$\omega<0$	accelerate_right	right	decelerate_right

A common issue for such a forward simulation is that it is almost impossible to exactly reach the goal configuration. In Hybrid A* search [17], an analytical expansion with the Reeds-Shepp method is responsible for this last step. However, as no such method is applicable under kinodynamic constraints, the search will finish when arriving in the neighborhood of q_{goal}.

IV. EXPERIMENTAL RESULTS

Experiments have been designed to verify and compare this method with RRT and a plain Hybrid A* search. The planning progress and result are visualized in a graphic interface as already illustrated in Fig. 1. The vehicle geometry is modeled as a rectangle, with an arrow indicating the forward direction. The reference point is at the middle of the rear axis. The start configuration is colored in green and the goal configuration in red. The obstacles are modeled with rectangles in gray. The yellow circles and lines are generated during the planning, with the resulting circle-path and trajectory displayed in blue.

The RRT is implemented after [20] and takes the same primitives as the heuristic search. The nearest node to the random sample is chosen in each iteration for expansion and a goal-bias is used with a specified probability. The Hybrid A* search plans in a grid of (x, y, θ) with constant resolution and considers only the kinematic constraints. The path optimization and trajectory generation parts from [17] are skipped to reduce the implementation effort.

All algorithms are implemented in C++ with the same vehicle kinematics, collision detection and visualization framework. The simulations run on a machine with 2.9 GHz CPU and 8 GB RAM. Each scenario generates 20 different q_{start} and q_{goal} by adding random displacements. The number of iterations are limited to 10 000, therefore a success-rate is counted. The Space Exploration Guided Heuristic Search is abbreviated as SEHS in the following sections.

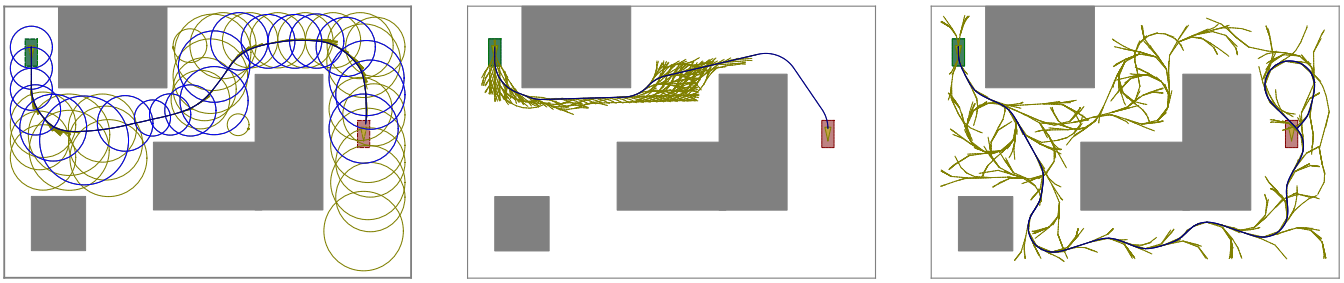
A. Low-Speed Navigation Scenario

The first scenario is a navigation problem through multiple obstacles in low-speed of $v_{\max} = 3\,\text{m}\,\text{s}^{-1}$, as shown in Fig. 2. The size of the scene is $60\,\text{m} \times 40\,\text{m}$. v_{start} and v_{goal} are both $0\,\text{m}\,\text{s}^{-1}$. The vehicle can accelerate with $a_{\max} = 1\,\text{m}\,\text{s}^{-2}$.

The results are listed in Table II. The number and time cost of the SEHS circle expansion are listed in parentheses. Compared to the total time cost, the space exploration takes a relative small portion of the total planning duration.

SEHS is comparable with Hybrid A* in time performance and both of them are one magnitude better than RRT. The Hybrid A* plans only with the vehicle kinematic constraints without optimization or trajectory generation. As mentioned in [17], the raw path from Hybrid A* search tends to "hug" the obstacles to achieve the minimum path distance, which is suboptimal for safety distances. In contrast, SEHS directly generates a trajectory with acceleration and steering commands. As a result from the heuristic design, the result trajectory tends to go through the circle centers, which stays away from the obstacles in all directions. Another obvious character of Hybrid A* search is the analytical expansion with the Reeds-Shepp method in the final stage. As can be seen from Fig. 2(b), this procedure saves almost one third of the total search range. SEHS needs to search until it reaches the close neighborhood of the goal position. Even so, SEHS takes takes only 10 % more nodes to finish the planning, which means that with the guidance from space exploration, the heuristic search of SEHS is more efficient than Hybrid A*.

The RRT, which is also a kinodynamic planner, and directly generates a trajectory, has the longest planning time and is not always able to solve the problem in the given limit. The success-rate is only 85 % and the result



(a) SEHS (kinodynamic planning) (b) Hybrid A* (path planning) (c) RRT (kinodynamic planning)

Fig. 2. Low-speed navigation scenario: $v_{\max} = 3\,\mathrm{m\,s^{-1}}$, $a_{\max} = 1\,\mathrm{m\,s^{-2}}$, $\phi_{\max} = 0.6\,\mathrm{rad}$, $\omega_{\max} = 0.6\,\mathrm{rad\,s^{-1}}$. The results from SEHS and RRT are trajectories with directly executable control inputs. The result from Hybrid A* is a path without speed and steering configurations.

varies drastically in time performance and path quality. Most result paths contain unnecessary motions, that still need to be optimized, as depicted in Fig. 2(c). It also has the lowest nodes expansion efficiency, which generates about 3000 nodes per second, compared with about 30 000 nodes per second from SHES and Hybrid A*. The reason is that the random samples are not always optimal to expand, and the overhead of the nearest neighbor search grows with the number of nodes.

TABLE II

RESULTS OF THE LOW-SPEED NAVIGATION SCENARIO

Planner	Fig.	%	nodes	time in ms
SEHS	2(a)	100	2858 (1595)	93.10 (4.15)
Hybrid A*	2(b)	100	2615	80.25
RRT	2(c)	85	4213	1361.20

B. High-Speed Overtaking Scenario

Another scenario is a high-speed overtaking problem. The vehicle has a initial speed of $20\,\mathrm{m\,s^{-1}}$, and can accelerate up to $30\,\mathrm{m\,s^{-1}}$ with a maximum acceleration of $5\,\mathrm{m\,s^{-2}}$. The driving corridor is 80 m long and 7 m wide, which can be divided into two lanes. A segment of the right lane is blocked by other traffic. The vehicle should change its lane to the left first to drive past the long obstacle, then steer back to the right lane to reach the goal position. The whole scenario is similar to general overtaking situations. The experimental results are shown in Fig. 3 and Table III.

TABLE III

RESULTS OF THE HIGH-SPEED OVERTAKING SCENARIO

Planner	Fig.	%	nodes	time in ms
SEHS	3(a)	100	331 (1528)	10.15 (2.95)
Hybrid A*	3(b)	100	182	10.10
RRT	3(c)	35	8173	1126.2

In this scenario setting, SEHS and Hybrid A* generate results in a rather short time frame and again outperform RRT. Their average planning duration of 10 ms satisfies the online requirement of most autonomous driving systems.

The shortage of kinematic path planners is obvious in this case. As the steering radius is always the same in Hybrid A*, the vehicle chooses to make a sharp turn just before the obstacle, which is dangerous or even impossible when driving in high-speed. It can be argued that the steering radius can be adapted to the vehicle speed during the planning. However, as kinematic path planners do not consider acceleration and steering speed, this adaptation is hard to carry out when the vehicle can dramatically change these two states. In the other case, SEHS provides a much more convenient trajectory. The steering radius is obvious larger than low-speed scenario, and the whole motion is smoother than the result from Hybrid A*.

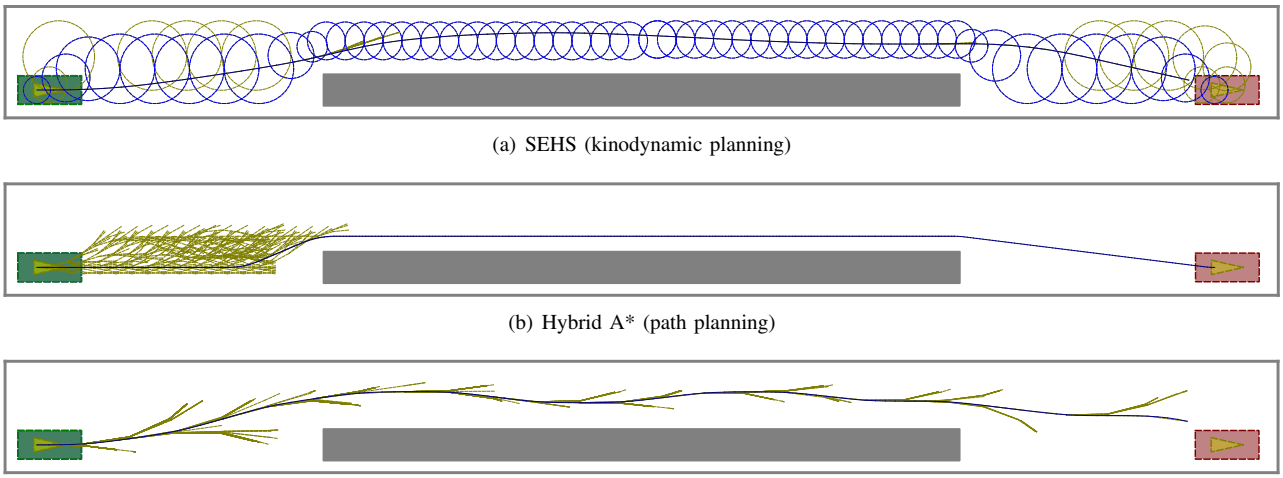
Another issue of the Hybrid A* result is the safety distance. As the Hybrid A* takes the shortest path under holonomic constraints as one heuristic, the planner produces a path closest to the obstacle due to the shorter distance. Further optimization is necessary to achieve a larger safety margin for the motion with distance to obstacles in the cost function. However, this feature is embodied in SEHS by the circle-path based heuristic, which guides the search expansions through the circle-centers.

The trajectory from SEHS possesses a slight discrepancy from the ideal form, which should travel exactly through the circle-centers during the overtaking phase. The reason is a suboptimal step size. As a result, the planner is not always able to drive the vehicle exactly to the middle of the lane, which needs a little effort to converge. Such behavior can be improved by adding a potential-based controller afterwards to push the vehicle away from the lane sides. Another solution could be gradually reducing the step size in a following incremental planning procedure to find the optimum, similar to RRT*.

The success-rate of RRT drops greatly in this scenario. The narrow passage between start and goal is extremely hard for sampling-based methods, as the sampling points are less likely to fall in the narrow passage, which hinders the tree to grow through the passage.

V. CONCLUSION AND FUTURE WORK

The Space Exploration Guided Heuristic Search method combines two simple, generic algorithms to enable efficient kinodynamic planning for nonholonomic vehicles. The space



(a) SEHS (kinodynamic planning)

(b) Hybrid A* (path planning)

(c) RRT (kinodynamic planning)

Fig. 3. High-speed overtaking scenario: $v_{\max} = 30\,\mathrm{m\,s^{-1}}$, $a_{\max} = 5\,\mathrm{m\,s^{-2}}$, $\phi_{\max} = 0.6\,\mathrm{rad}$, $\omega_{\max} = 0.6\,\mathrm{rad\,s^{-1}}$. The results from SEHS and RRT are trajectories with directly executable control inputs. The result from Hybrid A* is a path without speed and steering configurations.

exploration takes circles to explore the free space in a tree-like fashion, while the heuristic search uses the circle-path as heuristics to generate motions under kinodynamic constraints. This heuristic not only makes an estimation in time cost of the rest motion, but also provides information about search step size and posses the property to achieve optimal safety margin to obstacles. The performance of this approach is verified in several experiments with excellent results in both low-speed or high-speed scenarios.

The further development of SEHS will concentrate on dynamic scenarios. By knowing the speed, acceleration, steering and time cost of each nodes, SEHS can plan motions in more complicated scenarios involving moving obstacles. With the benefit of extremely efficient circle-based space exploration, it is possible to do replanning or adaptation in response to changes in the environment. The step size of the searching can also be continuously optimized in a incremental planning fashion.

Acknowledgements

This work is partially funded by the German Federal Ministry of Economics and Technology under grant no. 01ME12009 through the project RACE[1].

References

[1] J.-P. Laumond, *Robot Motion Planning and Control*. Springer, 1998.

[2] 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.

[3] 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.

[4] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. IEEE International Conference on Robotics and Automation*, 5 1993, pp. 802–807.

[5] 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*, 5 2001, pp. 1469–1474.

[1] http://www.projekt-race.de/

[6] N. Vandapel, J. Kuffner, and O. Amidi, "Planning 3-d path networks in unstructured environments," in *Proc. IEEE International Conference on Robotics and Automation*, 4 2005, p. 46244629.

[7] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

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

[9] S. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Department of Computer Science, Iowa State University, Tech. Rep., 1998.

[10] J. Kuffner Jr. and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. IEEE International Conference on Robotics and Automation*, 4 2000, pp. 995–1001.

[11] M. Kalisiak and M. van de Panne, "Rrt-blossom: Rrt with a local flood-fill behavior," in *Proc. IEEE International Conference on Robotics and Automation*, 5 2006, pp. 1237–1242.

[12] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *Proc. IEEE International Conference on Robotics and Automation*, 5 2009, pp. 2859–2865.

[13] E. Plaku, L. Kavraki, and M. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning," in *Proc. of Robotics: Science and Systems*, 6 2007.

[14] M. Rickert, O. Brock, and A. Knoll, "Balancing exploration and exploitation in motion planning," in *Proc. IEEE International Conference on Robotics and Automation*, 5 2008, pp. 2812–2817.

[15] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *Proc. IEEE International Conference on Robotics and Automation*, 5 2011, pp. 1478–1483.

[16] 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.

[17] 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.

[18] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2012.

[19] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: Towards real-time planning for robots with many degrees of freedom," Rice University, Houston, TX, USA, Tech. Rep. TR00-367, Aug. 2000.

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