2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Faster Collision Checks for Car-like Robot Motion Planning

Benjamin C. Heinrich, Dennis Fassbender and Hans-Joachim Wuensche

***Abstract*— In this paper, we describe how collision checking for car-like robots can be sped up utilizing system knowledge. Their non-holonomic motion, while being a challenge for motion planning, is utilized here to place discs which are used as an approximation of the robot's shape in a predictive manner. For ease of comparison, we assume the robot to be rectangular, i. e., we use bounding boxes.**

Our algorithm is compared to a widely-used baseline and shows similar performance in terms of under- and oversampling while being approximately 20–40 % faster. Another feature of the algorithm is its predictive nature: with the frontal disc, we already check for collisions that would occur with the rear disc in the next sample, assuming near-constant curvature. While this might be conservative in some cases where large steering rates are necessary, in our evaluation even tight corridors could be navigated without negative effects.

I. Introduction

When a robot, car-like or other, navigates the environment, one major condition is always to not collide with obstacles. Here, obstacles denote any static objects (e. g., walls, trees), untraversable ground (e. g., holes, quicksand) as well as dynamic objects (e. g., cars, humans). As pointed out, for instance, by [1], the computation of collision avoidance is still a major burden in motion planning.

Car-like robots move non-holonomically, i. e., their future movement is largely constrained by their current state. This poses a challenge for motion planning, but also yields possibilities for performance optimization when checking for collisions, as will be shown in this paper.

In addition to its movement constraints, the robot's shape is a key component for collision checking. Here, we focus on bounding boxes as they are a good first estimate for the shape of a car-like robot.

In order to obtain the area which is to be checked for collision, a given trajectory is usually sampled. A polygon is then built by joining the robot's shapes which are placed at the sampled points. This is done since the analytic description of the area given the trajectory is usually infeasible. A comparison between the planned trajectory and the actual area that needs to be checked – we will refer to this as the *occupancy* – can be seen in Figure 1.

There are different approaches to speed up collision checking. In [1], the computational power of the massive parallelization provided by graphic cards is utilized to pre-calculate obstacle grids such that a collision check is only a matter of a table look-up. The overhead of the sampling and data transfer, e. g., begins to pay off at about 28.8 K collision checks, for 72 angles (i. e., 2.5° discretization, utilizing symmetry.

All authors are with the Institute for Autonomous Systems Technology (TAS) of the Universität der Bundeswehr Munich, Neubiberg, Germany. Contact author email: benjamin.heinrich@unibw.de

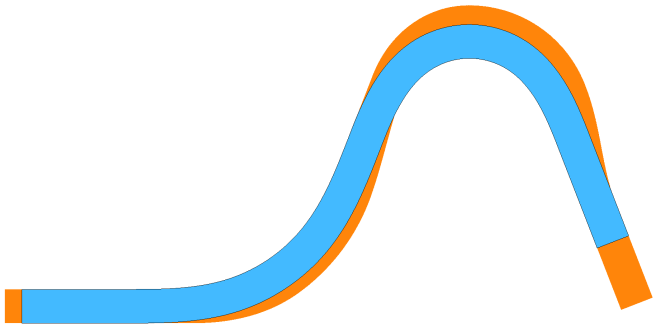


Fig. 1: Comparison of a planned trajectory in rear-axle coordinates (blue, shown with rear-axle width) and the joined polygon of bounding boxes sampled every cm, i. e., the actual occupancy (orange). Note especially the areas to left and right of the rear-axle path which grow with the trajectory's curvature.

Another method that does not use dilations but focuses also heavily on pre-calculation was introduced in [2]. There, a lookup table for a large set of relative (x, y)-positions, absolute yaw angles and different lengths and widths is constructed assuming rectangular vehicle shapes and slightly overestimating them. Here, obstacle checking, again, boils down to a table look-up. However, this method is only suited for checking collisions between instances of rectangles, i. e., one look-up for every traffic participant has to be done and environmental obstacles are not covered.

Here, we consider [3] as a our baseline method. There, the robot's shape is approximated by disc shapes, which lend themselves very well to collision checking in a distance-transformed obstacle grid. Discs are rotationally invariant and hence multiple pre-calculations of the whole grid are not necessary.

The baseline method was not only successfully used by the authors themselves in real competition (see [4]), but it is also in use by other known institutes (see [5]) and OEMs (see [6]). Classic methods, where only one circle is used for collision checking (see [7]) are a subset of the baseline method. It, however, also covers cases with a much closer shape approximation, as e. g., for parking maneuvers (eight discs were used in [8]).

This paper is structured as follows. First, we explain the baseline algorithm in section II. Then, we propose a faster method in section III. Both methods are compared in a number of scenarios in section IV before a conclusion is given in section V.

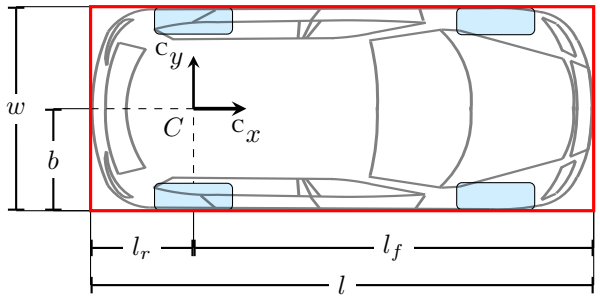


Fig. 2: A car-like robot with its bounding box (red, length l, width w). At the center of rotation, there is the planning frame C. Dependent key geometric dimensions are: length to rear, length to front and half width, denoted l_r, l_f and b, respectively. We assume C to be centered regarding w.

II. Baseline Collision Check

First, we present the collision-checking method from [3] as baseline. It considers bounding boxes and approximates their shape using discs. A bounding box has two main parameters, its length l and width w (see Figure 2). Note that, for car-like robots, a better approximation than a rectangular box might be desired and circles provide this to some extent. This is done by utilizing the discs' curvature to match the vehicle's contour better (see [9]). However, since this is equivalent to using a smaller bounding box, we will proceed using the baseline as is.

Obstacle checking is performed by testing the distance-transformed obstacle grid at the disc centers' location against their respective radii. The baseline algorithm is parameterizable by the positive uneven number $n \in \mathbb{N}$ of discs. Given n, the discs' radius r and distance constant d are defined as

$$r = \frac{1}{2}\sqrt{\frac{l^2}{n^2} + w^2}$$
$$d = 2\sqrt{r^2 - \frac{w^2}{4}}\,.$$

The disc centers are then placed relative to the bounding-box center at $i \cdot d$ along the robot's longitudinal axis, where $i \in \left\{-\left\lfloor \frac{n}{2} \right\rfloor, \ldots, \left\lfloor \frac{n}{2} \right\rfloor\right\}$. In the center-of-rotation frame C, where motion planning usually takes place, the disc center $\mathbf{d}$ is then given by

$$^{C}\mathbf{d}_{i,[x]} = \frac{l}{2} + l_r + i \cdot d$$
$$^{C}\mathbf{d}_{i,[y]} = 0\,,$$

where l_r is the length from $\mathbf{d}$ to the rear end of the bounding box. Note that l_r is negative in C. A depiction for $n \in \{3, 5\}$ can be found in Figure 3.

III. Faster Collision Check

Our main assumption is that a car-like robot moves momentarily on a circle with radius r_m. The overall traversed ground can then – momentarily – be described by an inner and an outer circle with radii

$$r_i = r_m - b \quad \text{and} \quad r_o = \mathrm{hypot}(r_m + b, l_f)\,,$$

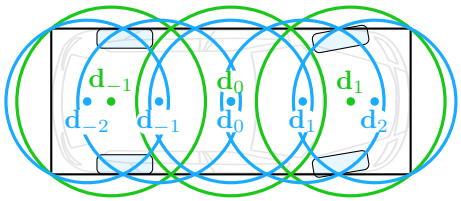


Fig. 3: Baseline algorithm for the case of $n \in \{3, 5\}$ discs, shown in green and blue, respectively. Note that for one sample the bounding box (black) is always completely covered. The lateral overestimation shrinks with n, the longitudinal increases.

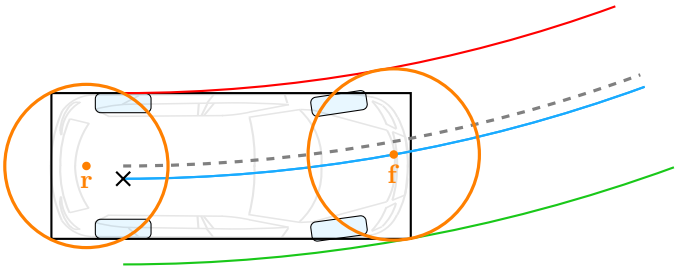


Fig. 4: Proposed algorithm, discs are shown in orange. An inner (red) and outer (green) circle describe the area to be checked for the vehicle's center of rotation traveling along the dashed trajectory. The frontal proposed disc is centered on the (blue) central circle such that the out-most point of the contour is covered. The rear proposed disc covers the rear bounding-box corners and the inner circle.

respectively (see: Figure 4, red and green lines). In the kinematic case, the inner circle is tangential to the vehicle chassis at its rear axle, the outer circle has the same center and goes through the outer front corner of the vehicle's bounding box.

Given the inner and outer circle, we find a central circle with radius

$$r_c = \frac{r_o + r_i}{2}$$

(see: Figure 4, blue line). There, we propose to place the frontal collision-checking disc center $\mathbf{f}$ with radius

$$r_p = \frac{r_o - r_i}{2}$$

at angle

$$\alpha = \arctan\left(\frac{l_f}{r_m + b}\right)\,,$$

along the central circle, starting at the negative y-axis (the point marked with a cross in Figure 4), resulting in

$$^{C}\mathbf{f}_{[x]} = r_c \sin(\alpha)$$
$$^{C}\mathbf{f}_{[y]} = b - r_p + r_c(1 - \cos(\alpha))\,. \tag{1}$$

Here, we can apply the trigonometric identities

$$\sin(\arctan(x)) = \frac{x}{\sqrt{1 + x^2}}$$
$$\cos(\arctan(x)) = \frac{1}{\sqrt{1 + x^2}} \tag{2}$$

for significant performance improvements.

In addition to this frontal disc, a second disc is placed at $\mathbf{r}$, covering the robot's rear end. Its radius is chosen such that it includes the rear corners of the bounding box as well as the inner circle, i. e.,

$$r_r = \mathrm{hypot}\left(b, \frac{l_r}{2}\right) .$$

It is placed at

$$\begin{aligned} {}^C\mathbf{r}_{[x]} &= l_r + \sqrt{r_r^2 - b^2} \\ {}^C\mathbf{r}_{[y]} &= 0 . \end{aligned} \tag{3}$$

A. Going Straight

Note that (1) is not defined for driving exactly straight, as the circle radii tend to infinity. Hence, when $r_m > 10000$, we proceed analogously to (3) and define

$$\begin{aligned} {}^C\mathbf{f}_{s,[x]} &= l_f - \sqrt{r_r^2 - b^2} \\ {}^C\mathbf{f}_{s,[y]} &= 0 . \end{aligned} \tag{4}$$

B. Tuning

Like the baseline method, our method also has a tuning factor $s' \in \mathbb{R}, s' \geq 1$. Here, s' directly scales the disc radii. Additionally, the disc centers are moved such that the outer corners of the bounding box remain exactly on the edge of the respective disc.

The proposed frontal disc's center remains on the central circle, but its angle α changes to $\alpha' = \alpha - \Delta\alpha$, where

$$\begin{aligned} \Delta\alpha &= \arccos\left(\frac{r_c^2 + (r_c + r_p)^2 - r_p^2 s^2}{2r_c(r_c + r_p)}\right) \\ &= \arccos\left(1 + \frac{r_p^2(1 - s^2)}{2r_c(r_c + r_p)}\right) \end{aligned}$$

by the law of cosines. The frontal disc center coordinates (1), are thus

$$\begin{aligned} {}^C\mathbf{f}'_{[x]} &= r_c \sin(\alpha') \\ {}^C\mathbf{f}'_{[y]} &= b - r_p + r_c(1 - \cos(\alpha')) . \end{aligned}$$

Recalling the structure of

$$\alpha' = \arctan\left(\frac{l_f}{r_m + b}\right) - \arccos\left(1 + \frac{r_p^2(1 - s^2)}{2r_c(r_c + r_p)}\right) ,$$

we can, analogously to (2), apply the trigonometric identities

$$\begin{aligned} \sin(\arctan(x) - \arccos(y)) &= \frac{xy - \sqrt{(1 - y^2)}}{\sqrt{1 + x^2}} \\ \cos(\arctan(x) - \arccos(y)) &= \frac{y + x\sqrt{(1 - y^2)}}{\sqrt{1 + x^2}} \end{aligned}$$

for significant performance improvements.

The rear proposed disc's center is shifted along the x-axis such that it always exactly includes the rear corners of the bounding box. Modifying (3) and (4), this leads to

$$\begin{aligned} {}^C\mathbf{r}'_{[x]} &= l_r + \sqrt{s'^2 r_p^2 - b^2} \\ {}^C\mathbf{f}'_{s,[x]} &= l_f - \sqrt{s'^2 r_p^2 - b^2} . \end{aligned}$$

For notational convenience, we define $s = 1000(s' - 1)$, where $s \geq 0, s \in \mathbb{R}$. A comparison for $s = 0$ and $s = 10$ can be seen in Figure 5.

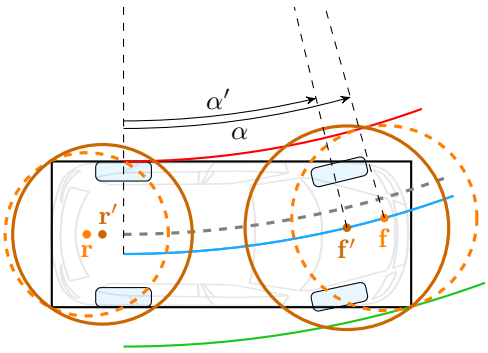


Fig. 5: Visualization of the influence of the tuning factor s. The base case $s = 0$ is shown dashed, the case for $s = 10$ solid. Note the shifted circle centers.

C. Discussion

With the proposed disc placement and radii, we try to cover the occupancy with only two discs. The computation time for the proposed disc-center placement is approximately the same as for the 5-disc baseline case due to the more complicated formulas. Nevertheless, this is compensated for by the lower number of actual grid evaluations that have to be carried out.

Additionally, the proposed approach is intended to yield better results when incrementally building a trajectory. This is due to the fact that the frontal disc already covers the area a baseline rear disc would cover on one of the next samples, assuming near-constant curvature. I. e., we already check potentially dangerous cells beforehand.

While this is conservative for trajectories with decreasing absolute curvature, it usually helps to speed up the search measurably by reducing the number of expansions. Note that checking the rear disc is indeed necessary as especially in cases with increasing absolute curvature the back of the car would not be covered sufficiently. Generally, the errors due to the constant-curvature assumptions are comparably small, as is shown in Section IV.

In very tight scenarios, especially when forward-backward maneuvers are necessary, we propose using a hierarchical approach where tighter discs are checked, should the search using the proposed method not yield any result. This is assumed feasible here, since high curvature changes at limited steering rates require lower speeds, and lower speed means more planning time is available. This effect is even more pronounced when changing the driving direction.

The proposed method is not suited for checking single positions, as the shape's center is not covered. This, however, is in most cases not a problem for planning trajectories as it will be checked by later samples. In Figure 6, we show that the trajectory is actually completely covered after 3 m. This, of course, is dependent on the geometric parameters of the bounding box and the tuning factor s.

The proposed method is tailored for *normal-sized* car-like robots driving comfortably. As the requirements and safety margins depend strongly on the scenario, quality of

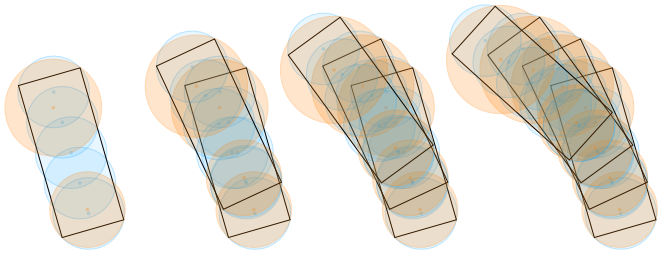


Fig. 6: Validity for proposed method with very short trajectories. The car-like robot makes a sharp left turn, starting at a curvature of 0.140 $\frac{1}{\mathrm{m}}$ finishing at 0.219 $\frac{1}{\mathrm{m}}$ within 4 m. The baseline method is shown for $n = 5$ case in blue, the proposed for $s = 5$ in orange, the sampling distance δ was set to 1 m. From left to right one to four samples are evaluated.

environment representation etc., no general evaluation was carried out. Instead, we give results regarding geometry, scenarios and environment representations as we experience them with our vehicles in off-road scenarios. Nevertheless, the tuning should be intuitive and extensions for, e. g., extra long vehicles by adding an additional disc are possible.

Note that driving backwards was not explicitly covered here but works analogously.

IV. Results

For the evaluation of the algorithms, let us first introduce some performance metrics. For this, let true area to be checked and its estimates be denoted A and $\hat{A}_j$, respectively. Furthermore, as checking is often done in discretized grids, let ΔA and $\Delta\hat{A}$ denote their 0.2 m-rastered version.

- *Oversampling* describes the area covered by discs, but not the occupancy. For algorithm j it is $O_j \triangleq \hat{A}_j \setminus A$. Analogously for ΔO_j.
- *Undersampling* describes the area covered by the occupancy, but not the discs. For algorithm j it is $U_j \triangleq A \setminus \hat{A}_j$. Analogously for ΔU_j.
- *Computation Time* comprises the calculation of the disc centers and the evaluation of the obstacle distances. For algorithm j it is denoted T_j and given in µs.

Hence, undersampling is a measure of the danger of collision and should consequently be kept as close to zero as possible. Oversampling on the other hand gives the conservativeness of the algorithm and is thus a measure of performance (smaller is better) in complex environments.

All results were produced on an Intel i7-3770K CPU 3.50 GHz with 24 GB of RAM. The code was written in C++ and compiled with GCC 8.1.1. No parallelization was used. All evaluation times include a logging overhead. Nevertheless, since both methods are affected the same way, the relative speed-up is not compromised.

The geometric parameters were taken from our research vehicle MuCAR-3 (Munich Cognitive Autonomous Robot, 3rd Generation) and are

$$l = 4.754\,\mathrm{m} \qquad l_f = 3.781\,\mathrm{m}$$
$$w = 1.928\,\mathrm{m} \qquad l_r = -0.973\,\mathrm{m}\,.$$

The test vehicle is a SUV, hence its geometric parameters are representative for a car-like robot.

A. Generic Test

For a generic test, we use the trajectory shown in Figure 1. It consists of eight 6 m parts with linear change in curvature from

$$0 \to 0 \to .1 \to .1 \to 0 \to -.2 \to -.2 \to 0 \to 0\,.$$

I. e., there are always two straight, constant-curvature, increasing and decreasing curvature clothoidal segments. The performance metrics can be seen in Figure 8. Note that we opted to neither pre-calculate the disc configurations for different trajectory-sample yaw angles nor, in the proposed case, for different curvatures.

An interesting effect is that going over $n = 5$ when using the baseline algorithm does not lead to proportionally longer computation times. This is mainly due to the fact that the actual distance checking per circle takes only approximately 60 ns and thus increasing the number of circles does not have a huge effect.

B. Application in Planning

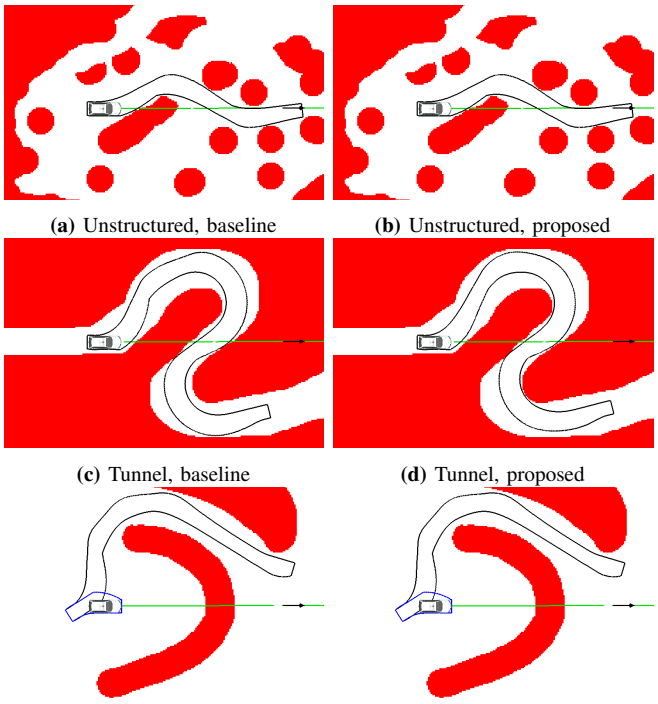
Until now, we always assumed to check a given trajectory. Another use case is to generate a trajectory incrementally, e. g., by an A* search. Thus, we implemented the proposed obstacle avoidance discs in our planning framework. For more details about it, please refer to [9], [10], [11].

We test three cases, depicted in Figure 7. The first scenario is in an unstructured environment. The second features a tight tunnel with an s-shape. In the last scenario, a dead end is handled.

The resulting trajectories for the baseline ($n = 5$) and proposed case are very similar, as can be seen visually or from the cost in Table I. As the distance to obstacles is also a factor in the calculation of the cost function, different discs necessarily mean different cost. However, the absolute costs are surprisingly even between both approaches, Note that in the tunnel case, were the proposed method has higher costs, the final solution comes 1.51 m closer to the goal pose.

We used 7 forward and potentially 3 backward expansions for all scenarios. In all cases, the total number of expansions is roughly 10 % lower for the proposed method. This is due to the predictive nature of the disc placement, i. e., branches that are highly likely to lead to collisions will not get expanded as long as there are more promising ones. The smaller number of expansion together with the lower computational cost per expansion result in a net speed gain of 35.3 %, 38.6 % and 28.7 % for the three respective scenarios.

The large total number of expansions is due to our planner being designed for sensor-based planning. I. e., application on an autonomous vehicle where obstacles are only detectable with direct line of sight and at a limited range. This means planner designed for large a-priori grid maps will most likely outperform the absolute numbers given in Table I. The relative performance gain, however, is not compromised by that.



(a) Unstructured, baseline **(b)** Unstructured, proposed

(c) Tunnel, baseline **(d)** Tunnel, proposed

(e) Dead end, baseline **(f)** Dead end, proposed

Fig. 7: Test cases. Obstacles are depicted red, a global path to follow in green. The contour of the planned trajectory is shown in black (or blue for backwards).

TABLE I: Test results for Figure 7

	collision (ms)	expansions	cost
	Unstructured		
baseline, $n = 5$	7.369	1197	46.822
proposed, $s = 0$	4.767	1062	46.586
	Tunnel		
baseline, $n = 5$	104.965	20 888	99.212
proposed, $s = 0$	64.471	18 196	99.256
	Dead end		
baseline, $n = 5$	165.556	29 128	73.028
proposed, $s = 0$	118.044	27 453	72.819

V. Conclusion

We proposed a method for faster collision checking in the context of motion planning. Our algorithm is explicitly designed for robots that momentarily move on a circle, e. g., car-like robots. We consider bounding boxes as they generalize car-like robots quite well. It should be noted, though, that it is often wise to shorten the bounding box for the discs to fit the actual shape better.[1]

The proposed method places a fixed number of only two discs in contrast to the well-known baseline algorithm [3], where their number n is a tuning factor. Our method also has a tuning factor s, which scales the trade-off between over- and undersampling of the true occupancy. Since the frontal disc of the proposed method already checks most of the area which will be covered by the vehicle contour assuming near-constant curvature it is well suited for exploring algorithms, such as A*. This property can be interpreted as being predictive.

The comparison to the baseline showed comparable performance in terms of under- and oversampling (for the case of $n = 5$ and $s = 5$) while being approximately as fast to calculate as the $n = 3$ case. It was also shown that even though being conservative due to the assumption of near-constant curvature, even very tight passages (see Figure 7d) can be handled. We were able to achieve this increased performance by using knowledge of the system's motion model.

[1]Car-like robots usually do not have a rectangular shape but have rounded corners, see fig. 2 and compare the car's shape to the bounding box or [9], for discs we actually used.

References

[1] G. Tanzmeister, M. Friedl, D. Wollherr, and M. Buss, "Efficient Evaluation of Collisions and Costs on Grid Maps for Autonomous Vehicle Motion Planning," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2249–2260, Oct 2014.

[2] A. Rizaldi, S. Söntges, and M. Althoff, "On Time-Memory Trade-Off for Collision Detection," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, June 2015, pp. 1173–1180.

[3] J. Ziegler and C. Stiller, "Fast Collision Checking for Intelligent Vehicle Motion Planning," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, San Diego, CA, USA, 2010, pp. 518–522.

[4] A. Geiger, M. Lauer *et al.*, "Team AnnieWAY's Entry to the 2011 Grand Cooperative Driving Challenge," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1008–1017, Sept 2012.

[5] F. Kunz, D. Nuss *et al.*, "Autonomous Driving at Ulm University: A Modular, Robust, and Sensor-Independent Fusion Approach," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, June 2015, pp. 666–673.

[6] B. Gutjahr, L. Gröll, and M. Werling, "Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1586–1595, June 2017.

[7] Z. Qu, J. Wang, and C. E. Plaisted, "A New Analytical Solution to Mobile Robot Trajectory Generation in the Presence of Moving Obstacles," *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 978–993, Dec 2004.

[8] S. Klaudt, A. Zlocki, and L. Eckstein, "A-priori Map Information and Path Planning for Automated Valet-Parking," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, June 2017, pp. 1770–1775.

[9] D. Fassbender, A. Mueller, and H.-J. Wuensche, "Trajectory Planning for Car-Like Robots in Unknown, Unstructured Environments," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, Chicago, IL, USA, Sep. 2014.

[10] D. Fassbender, B. C. Heinrich, and H.-J. Wuensche, "Motion Planning for Autonomous Vehicles in Highly Constrained Urban Environments," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, Daejeon, South Korea, Oct. 2016.

[11] H. Jaspers, M. Himmelsbach, and H.-J. Wuensche, "Multi-modal Local Terrain Maps from Vision and LiDAR," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, Redondo Beach, CA, USA, Jun. 2017.

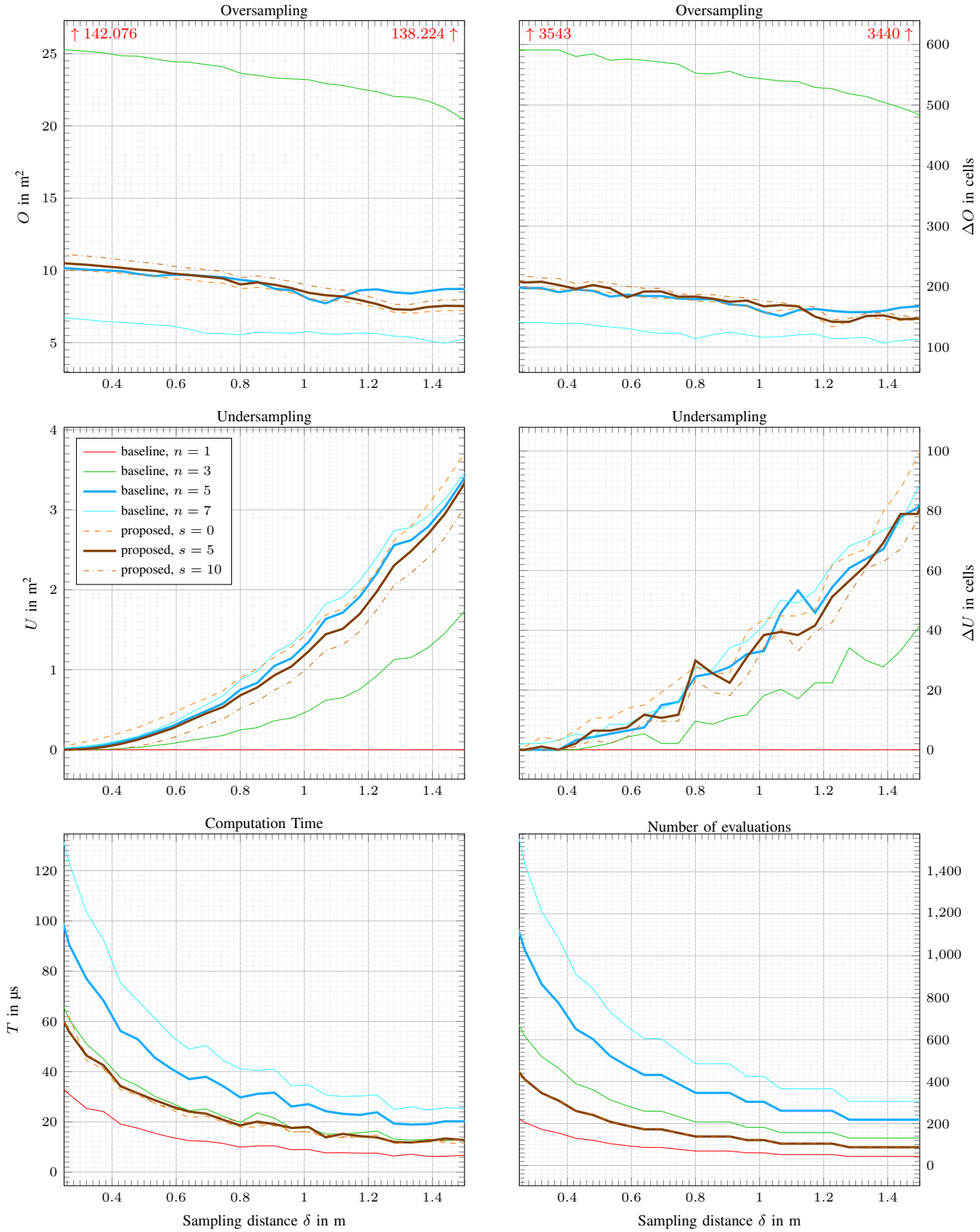


Fig. 8: Results regarding Figure 1; computation times were averaged over 30 runs. Note that additionally, a distance transform of the obstacle grid is calculated, which takes about 0.63 ms on an 60 m × 60 m grid with 0.2 m resolution. The actual checking then takes about 60 ns per circle center.