IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 5, 2021, accepted June 22, 2021, date of publication June 29, 2021, date of current version July 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3093471

Probabilistic Robust Path Planning for Nonholonomic Arbitrary-Shaped Mobile Robots Using a Hybrid A* Algorithm

TOBIAS RAINER SCHÄFLE AND NAOKI UCHIYAMA
Department of Mechanical Engineering, Toyohashi University of Technology, Toyohashi 441-8580, Japan
Corresponding author: Tobias Rainer Schäfle (tobias.schaefle@gmx.de)

This work was supported in part by Toyometal Company Ltd., Toyohashi, Japan, and in part by the Toyohashi University of Technology, Japan, as one of the Cooperative Projects for Innovative Research.

This work did not involve human subjects or animals in its research.

ABSTRACT To guarantee safe motion planning, the underlying path planning algorithm must consider motion uncertainties and uncertain state information related to static, and dynamic obstacles. This paper proposes novel hybrid A* (HA*) algorithms that consider the uncertainty in the motion of a mobile robot, position uncertainty of static obstacles, and position and velocity uncertainty of dynamic obstacles. Variants of the HA* algorithm are proposed wherein a soft constraint is used in the cost function instead of chance constraints for probability guarantees. The proposed algorithm offers a tradeoff between the traveling distance and safety of paths without pruning additional nodes. Furthermore, this paper introduces a method for considering the shape of a mobile robot for probabilistic safe path planning. The proposed algorithms are compared with existing path planning algorithms and the performance of the algorithms is evaluated using the Monte Carlo simulation. Compared with the related probabilistic robust path planning algorithms, the proposed algorithms significantly improved safety without excessively increasing travel distance and computational time. The results also showed that dynamic obstacles were safely avoided, which is in contrast to the conventional HA* algorithm that has a high probability of collision. In addition, considering the shape of the robot in the proposed probabilistic approach led to safer paths overall.

INDEX TERMS Dynamic obstacle, nonholonomic mobile robot, path planning, probabilistic safety.

I. INTRODUCTION

Self-driving cars face with many challenges in terms of perception, localization, and control. A key difficulty in designing an autonomous vehicle is path planning. The vehicle is required to find a feasible path from its starting pose to the desired goal pose without collision. For this demand, many different path planning algorithms have been developed (e.g., [1]–[3]).

A new graph-search-based path planner called hybrid A* (HA*) was introduced in the DARPA Urban Challenge in 2007 [2], [4], which was modified in [5] for multiple, subsequent goal poses. The HA* algorithm is based on the A* path planner [6]. However, unlike A*, the HA* algorithm has expanded its nodes with continuous vehicle coordinates, thus guaranteeing feasible paths. In general, A*-based path planners are piecewise linear. Recent research has enabled the planner to re-plan [7], [8] or use any angle for the piecewise linear path [9]. Although these algorithms can find optimal and feasible paths to a desired goal position, none of them consider the uncertainty of the environment or robot. In particular, A*-based approaches tend to find paths that are very close to obstacles; hence, in such approaches, safety cannot be guaranteed. Some recent studies used reinforcement learning to teach mobile robots to move from a starting position to the desired goal position without colliding with obstacles [10], [11]. Sampling-based planners (SBP), particularly rapidly-exploring random tree (RRT) algorithms [12], [13], were modified to guarantee safety. The arrival field method using Eikonal equations was used in [14] to generate time optimal paths, where the robot moves slower near obstacles. Bry *et al.* introduced the rapidly-exploring random belief trees (RRBT) [15] algorithm, which uses local linear quadratic Gaussian (LQG) control solutions to predict

The associate editor coordinating the review of this manuscript and approving it for publication was Rui-Jun Yan.

distributions over trajectories. A particle-filter based RRT path planner (pRRT) was developed by Melchior and Simmons [16], where each extension of the search tree was simulated multiple times under different conditions. Additionally, nodes were created by clustering the simulation results. Blackmore *et al.* presented a probabilistic approach that used the value of the maximum probability of a robot colliding with an obstacle [17], [18]. The probability of collision is expressed as a disjunction of deterministic linear constraints. Luders *et al.* combined the chance-constrained method of Blackmore *et al.* with RRT (CCRRT) and extended it by including uncertainty for obstacles [19]. The CCRRT algorithm was then combined with a method for predicting future obstacle behavior in the planner [20]. Furthermore, for linear dynamics it was extended for the RRT* algorithm [21], which they termed CCRRT* [22]. Model predictive control was combined with the particle approach to achieve optimal robust solutions [23]. Particle-based approaches were introduced to guarantee probabilistic safe paths, which considered non-Gaussian uncertainty [24]. Van den Berg *et al.* proposed a variant of LQG for motion planning to consider motion uncertainty and imperfect state information [25]. A novel motion planner [26] was modified to consider perception and control uncertainties for self-driving vehicles in [27].

Existing works mainly extend sampling-based planners to enhance the probabilistic robustness, in which results for nonlinear mobile robot dynamics observed a "zig-zag" motion leading to large traveling distances [24]. By contrast, modified HA* path planning algorithms that consider the motion uncertainty of mobile robots and the uncertain state of static and dynamic obstacles are proposed in this work. They employ the chance constraints presented in [17] and [19] to guarantee probabilistic robustness. A disadvantage of the chance constraints for collision avoidance in the probabilistic robust path planners is the additional pruning of nodes. Especially in narrow environments, probabilistic robust approaches may be unable to find a path to a goal pose. Hence, a novel cost function that allows a tradeoff between traveled distance and safety is introduced in this work. Expanded nodes with a large probability of collision incur large cost penalties when not pruned. Furthermore, existing path planning models assume point or circular-shaped mobile robots. Extending probabilistic robustness for the shape of the robot is difficult due to the computational complexity of extending the configuration space. The proposed approach uses an approximation for arbitrary-shaped mobile robots in a deterministic environment [28] and extends it for state and environment uncertainties and is, therefore, able to consider probabilistic robustness for arbitrary-shaped mobile robots. This work considers linear dynamics and uncertainties in the position and velocity of dynamic obstacles. The contributions and motivations of this study are summarized as follows.

(i) Existing path planning models that consider uncertainties in the system states and environment use incremental sampling-based path planners (e.g., RRT). We believe this work presents the first HA*-based probabilistic robust path planning algorithm.
(ii) In this work, a soft constraint for the collision probability is employed, instead of chance constraints, which increases the search space of the algorithm by keeping candidate nodes that will be otherwise deleted by the chance constraints.
(iii) Other existing approaches consider probabilistic safety only for point or circular-shaped mobile robots. So far, to the best of our knowledge, probabilistic collision detection for arbitrary-shaped mobile robots has not been studied. A novel method to extend the state uncertainty of arbitrary-shaped mobile robots, which explicitly considers orientational uncertainties, is proposed in this study.

Section II presents the problem considered in this study followed by the HA* algorithm under Gaussian uncertainty in Section III. Section IV introduces the extension of the chance constraints for arbitrary-shaped mobile robots. In Section V the results are presented, and the effectiveness of the proposed HA* algorithms is demonstrated in different environments, and a performance comparison test of the algorithms is given using the Monte Carlo simulation (MCS). A discussion is given in Section VI and conclusions and future works are presented in Section VII.

NOTATION

$\mathbb{R}^n$ and $\mathbb{R}^{n\times m}$ represent an n-dimensional Euclidean space and $(n \times m)$-dimensional matrix, respectively; $\boldsymbol{A}^{\mathrm{T}}$ and I_n represent the transpose of the matrix $\boldsymbol{A}$ and n-dimensional identity matrix, respectively. Bold uppercase, bold lowercase, and lowercase letters indicate matrices, vectors, and scalars, respectively. Uppercase Σ indicates a covariance matrix; σ_x^2 indicates the variance of variable x; $\sigma_{x,y}$ indicates the covariance of variables x, y. The mean of a variable x is given by μ_x. $\boldsymbol{\epsilon_x} \sim \mathcal{N}(\boldsymbol{\mu_x}, \Sigma_{\boldsymbol{x}})$ denotes the random Gaussian process disturbance of state vector $\boldsymbol{x}$.

II. PROBLEM FORMULATION

This study considers a planar motion of arbitrary-shaped mobile robots whose configuration space $\mathcal{C} \in \mathrm{SE}(2) = \mathbb{R}^2 \times \mathrm{SO}(2)$ is the special Euclidean group which consists of two-dimensional position and orientation. Furthermore, the configuration space is divided into open space $\mathcal{C}_{\mathrm{free}} \subseteq \mathcal{C}$ and obstacles $\mathcal{C}_{\mathrm{obs}} \subset \mathcal{C}$, where $\mathcal{C}_{\mathrm{free}} + \mathcal{C}_{\mathrm{obs}} = \mathcal{C}$. In a deterministic environment, the objective is to find a path from the start node $\boldsymbol{x}_{\mathrm{S}} \in \mathcal{C}_{\mathrm{free}}$ to the goal node $\boldsymbol{x}_{\mathrm{G}} \in \mathcal{C}_{\mathrm{free}}$. The complete path $\mathcal{P} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_{t_{\mathrm{goal}}})$ is described by the set of nodes whose ends are the start and goal nodes, $\boldsymbol{x}_{\mathrm{S}} = \boldsymbol{x}_0$ and $\boldsymbol{x}_{\mathrm{G}} = \boldsymbol{x}_{t_{\mathrm{goal}}}$, respectively. Moreover, $\boldsymbol{x}_t$ is the parent node of $\boldsymbol{x}_{t+1}$, $t \in \mathbb{N}$ is the respective integer time instant, and t_{goal} is the final time instant. The optimization

problem is described as follows:

$$\begin{aligned} &\text{minimize} \sum_{t=1}^{t_{\text{goal}}} J(\boldsymbol{x}_t) \\ &\text{subject to } \boldsymbol{x}_t \in \mathcal{C}_{\text{free}} \quad \forall t \in \{0, \ldots, t_{\text{goal}}\}, \end{aligned}$$

where $J(\boldsymbol{x}_t)$ is the cost function. In belief space planning, the collision condition is described by chance constraints and the cost function is the expected cost of $J(\boldsymbol{x}_t)$. Accordingly, the stochastic optimization problem is described as

$$\begin{aligned} &\text{minimize} \sum_{t=1}^{t_{\text{goal}}} E[J(\boldsymbol{x}_t)] \\ &\text{subject to } p(\boldsymbol{x}_t \in \mathcal{C}_{\text{obs}}) \leq \Gamma \quad \forall t \in \{0, \ldots, t_{\text{goal}}\}, \end{aligned}$$

where $E[\cdot]$, $p(\cdot)$ and Γ denote the expected value, probability of the outcome, and threshold value for collision, respectively.

Static and dynamic obstacles in the configuration space are represented as convex polygons, as illustrated in Fig. 1, which allows a fast collision analysis and makes it possible to incorporate a probabilistic collision check [17]. It is assumed that each vertex of an obstacle is known, and the obstacles are obtained from an offline map. In the following section, the robot is assumed to be a point robot, and therefore collision will occur when the robot point is inside the obstacle. Using the known obstacle vertices and robot position $\boldsymbol{q}_t = [x_t, y_t]^{\mathrm{T}}$, the collision conditions are expressed as follows:

$$\bigwedge_{i=1}^{n_e} \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{q}_t < b_i, \tag{1}$$

where n_e is the number of edges and $\boldsymbol{a}_i$ and b_i denote the parameters of the straight-line equation $\boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{q}_t = b_i$ of edge i, respectively.

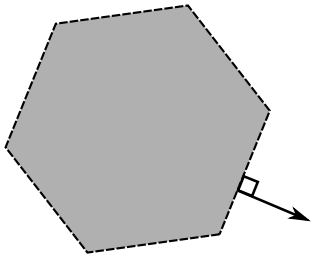


FIGURE 1. Convex obstacle representation in the configuration space.

III. HYBRID A* ALGORITHM UNDER GAUSSIAN UNCERTAINTY

This section describes the uncertainty propagation for a kinematic model of a nonholonomic mobile robot [29]. After briefly explaining the conventional HA* algorithm [4], this section introduces the collision condition for probabilistic safe motion. The linear velocity model of dynamic obstacles and their uncertainty propagation is introduced.

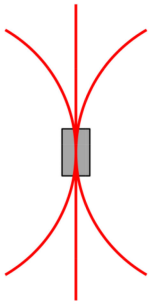


FIGURE 2. Example motions for HA* using the velocity model.

A. UNCERTAINTY PROPAGATION

In this work, the velocity model from [29] is used to model the kinematics of the nonholonomic mobile robot. The deterministic kinematic model of the form

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{2}$$

with nonzero angular velocity is as follows [29]:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} -\frac{v_t}{\omega_t} \sin\theta_t + \frac{v_t}{\omega_t} \sin(\theta_t + \omega_t \delta t) \\ \frac{v_t}{\omega_t} \cos\theta_t - \frac{v_t}{\omega_t} \cos(\theta_t + \omega_t \delta t) \\ \omega_t \delta t \end{bmatrix}, \tag{3}$$

where $\boldsymbol{x}_t = [x_t, y_t, \theta_t]^{\mathrm{T}} \in \mathrm{SE}(2)$ is the pose of the robot at time instant t with x_t, y_t being the position in the global reference frame and θ_t being the heading of the robot. Moreover, $\boldsymbol{u}_t = [v_t, \omega_t]^{\mathrm{T}}$, where v_t and ω_t are the linear and angular velocities, respectively, and δt is the sampling time. The model for straight motions can be obtained using L'Hôpital's rule for $\lim_{\omega_t \to 0} f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ as follows:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} v_t \delta t \cos\theta_t \\ v_t \delta t \sin\theta_t \\ 0 \end{bmatrix}. \tag{4}$$

Example motions using the velocity model for HA* are shown in Fig. 2. The additive Gaussian process noise is integrated into the model by replacing $\boldsymbol{u}_t = [v_t, \omega_t]^{\mathrm{T}}$ with $\hat{\boldsymbol{u}}_t = [\hat{v}_t, \hat{\omega}_t]^{\mathrm{T}}$, where

$$\begin{bmatrix} \hat{v}_t \\ \hat{\omega}_t \end{bmatrix} = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} + \boldsymbol{\varepsilon}_M \tag{5}$$

with

$$\boldsymbol{\varepsilon}_M \sim \mathcal{N}(0, \Sigma_M). \tag{6}$$

Here, Σ_M is the covariance matrix of the random variable $\boldsymbol{\varepsilon}_M$. The Gaussian probability distribution $\Sigma_{\boldsymbol{x}_t}$ is obtained using the prediction step of the extended Kalman filter [30]. Hence, the nonlinear kinematic model of the robot has to be linearized at each time instant around the input $\boldsymbol{u}_t$ and state mean $\boldsymbol{\mu}_{\boldsymbol{x}_t}$

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{x}_{t+1}} &\approx \tilde{\boldsymbol{A}}_t \boldsymbol{\mu}_{\boldsymbol{x}_t} + \tilde{\boldsymbol{B}}_t \boldsymbol{u}_t \\ \tilde{\boldsymbol{A}}_t &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{\mu}_{\boldsymbol{x}_t}, \boldsymbol{u}_t) \end{aligned}$$

$$\tilde{\boldsymbol{B}}_t = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\boldsymbol{\mu}_{\boldsymbol{x}_t}, \boldsymbol{u}_t), \tag{7}$$

where

$$\tilde{\boldsymbol{A}}_t = \begin{bmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{\theta_t} + \frac{v_t}{\omega_t}\cos(\mu_{\theta_t} + \omega_t\delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\mu_{\theta_t} + \frac{v_t}{\omega_t}\sin(\mu_{\theta_t} + \omega_t\delta t) \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$\tilde{\boldsymbol{B}}_t = \left[\tilde{\boldsymbol{b}}_1, \tilde{\boldsymbol{b}}_2\right]$$

$$\tilde{\boldsymbol{b}}_1 = \begin{bmatrix} \frac{-\sin\mu_{\theta_t} + \sin(\mu_{\theta_t} + \omega_t\delta_t)}{\omega_t} \\ \frac{\cos\mu_{\theta_t} - \cos(\mu_{\theta_t} + \omega_t\delta t)}{\omega_t} \\ 0 \end{bmatrix}$$

$$\tilde{\boldsymbol{b}}_2 = \begin{bmatrix} \frac{v_t(\sin\mu_{\theta_t} - \sin(\mu_{\theta_t} + \omega_t\delta t))}{\omega_t^2} + \frac{v_t\cos(\mu_{\theta_t} + \omega_t\delta t)\delta t}{\omega_t} \\ -\frac{v_t(\cos\mu_{\theta_t} - \cos(\mu_{\theta_t} + \omega_t\delta t))}{\omega_t^2} + \frac{v_t\sin(\mu_{\theta_t} + \omega_t\delta t)\delta t}{\omega_t} \\ \delta t \end{bmatrix} \tag{9}$$

for $\omega_t \neq 0$. For straight motions, the linearization is given as follows:

$$\tilde{\boldsymbol{A}}_t = \begin{bmatrix} 1 & 0 & -v_t\delta_t\sin\mu_{\theta_t} \\ 0 & 1 & v_t\delta_t\cos\mu_{\theta_t} \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$\tilde{\boldsymbol{B}}_t = \begin{bmatrix} \delta_t\cos\mu_{\theta_t} & 0 \\ \delta_t\sin\mu_{\theta_t} & 0 \\ 0 & 0 \end{bmatrix}. \tag{11}$$

With the linearized model, Gaussian uncertainty distribution $\Sigma_{\boldsymbol{x}_{t+1}}$ can be updated as follows:

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{x}_{t+1}} &= \boldsymbol{f}(\boldsymbol{\mu}_{\boldsymbol{x}_t}, \boldsymbol{u}_t) \\ \Sigma_{\boldsymbol{x}_{t+1}} &= \tilde{\boldsymbol{A}}_t\Sigma_{\boldsymbol{x}_t}\tilde{\boldsymbol{A}}_t^{\mathrm{T}} + \tilde{\boldsymbol{B}}_t\Sigma_M\tilde{\boldsymbol{B}}_t^{\mathrm{T}}, \end{aligned} \tag{12}$$

where $\tilde{\boldsymbol{B}}_t\Sigma_M\tilde{\boldsymbol{B}}_t^{\mathrm{T}}$ is the mapping of the motion noise from the control space to state space [29].

B. ALGORITHM OVERVIEW

The conventional HA* [4] is a variation of the A* algorithm [6]. Unlike the A* algorithm, the HA* algorithm assigns a continuous mobile robot coordinate to each discrete cell in the working space. Each node of the planner is expanded by applying three different actions: no-turn, left turn, and right turn for forward and reverse motions (see Fig. 2). Reverse motions are only applied if a forward motion violates a collision constraint. A kinematic model of the mobile robot is used to generate new states for the planner ((3) and (4)), and the same cost-to-goal heuristics as in [4] are applied: The *non-holonomic-without-obstacles* heuristic, which uses Reeds and Shepp paths [31] to compute the shortest path to the goal (assuming no obstacles) from every point in the working space, and the *holonomic-with-obstacles* heuristic, which employs the occupancy grid of the configuration space to compute the shortest path to the goal via dynamic programming. The maximum heuristic cost of both heuristics is used for the path planner.

For every N^{th} node, an expansion of the current node to the goal with Reeds and Shepp paths is generated, where N depends on the cost-to-goal heuristic. If the expansion does not collide with any obstacles, the HA* algorithm is completed.

The HA* algorithm uses a cost function to penalize the reverse motion and switching of the motion direction of the path traveled:

$$J(\boldsymbol{x}_t) = l_t(1 + r_t C_{\text{rev}}) + |r_t - r_{t-1}|C_{\text{sw}}, \tag{13}$$

where l_t is the distance traveled from time instant $t-1$ to time instant t, and C_{rev} and C_{sw} are the penalty gains for driving in reverse and switching the direction of motion, respectively.

This work modifies the HA* algorithm by considering Gaussian uncertainties in the robot's motion, as well as the static obstacle position. Furthermore, dynamic obstacles with uncertain state information are considered. Hence, the robot's uncertain state is updated at each node expansion using (12). If the condition

$$p(\boldsymbol{x}_t \in \mathcal{C}_{\text{obs}}) \leq \Gamma \tag{14}$$

is satisfied the current node is probabilistically safe and the expected cost

$$E[J(\boldsymbol{x}_t)] = \mu_{l_t}(1 + r_t C_{\text{rev}}) + |r_t - r_{t-1}|C_{\text{sw}} \tag{15}$$

is calculated, where μ_{l_t} is the mean distance traveled from time instant $t-1$ to time instant t. If the condition is not satisfied the node will be pruned.

C. STATIC AND DYNAMIC OBSTACLES

1) STATIC OBSTACLES

The static obstacles are expressed in a manner that is similar to [19]. The probability distributions of the static obstacles are time invariant, i.e.,

$$\underbrace{\begin{bmatrix} x_{\mathcal{S}^j} \\ y_{\mathcal{S}^j} \end{bmatrix}}_{\boldsymbol{x}_{\mathcal{S}^j}} = \underbrace{\begin{bmatrix} x_{\mathcal{S}_0^j} \\ y_{\mathcal{S}_0^j} \end{bmatrix}}_{\boldsymbol{x}_{\mathcal{S}_0^j}} + \boldsymbol{\varepsilon}_{\mathcal{S}^j} \tag{16}$$

with

$$\boldsymbol{\varepsilon}_{\mathcal{S}^j} \sim \mathcal{N}(0, \Sigma_{\mathcal{S}^j}) \quad j \in \{1, \ldots, n_{\mathcal{S}}\}, \tag{17}$$

where $[x_{\mathcal{S}^j}, y_{\mathcal{S}^j}]^{\mathrm{T}}$ is the position of obstacle j, $[x_{\mathcal{S}_0^j}, y_{\mathcal{S}_0^j}]^{\mathrm{T}}$ is the nominal position of obstacle j, and $n_{\mathcal{S}}$ is the number of static obstacles in the configuration space. Hereafter, all static obstacles have the same covariance, i.e., $\Sigma_{\mathcal{S}_j} = \Sigma_{\mathcal{S}}$.

2) DYNAMIC OBSTACLES

The dynamic obstacles are assumed to be convex with an estimated mean heading and velocity. This work uses a linear velocity model for the dynamic obstacle

$$\begin{bmatrix} x_{\mathcal{D}_{t+1}} \\ y_{\mathcal{D}_{t+1}} \\ \dot{x}_{\mathcal{D}_{t+1}} \\ \dot{y}_{\mathcal{D}_{t+1}} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x_{\mathcal{D}_t} \\ y_{\mathcal{D}_t} \\ \dot{x}_{\mathcal{D}_t} \\ \dot{y}_{\mathcal{D}_t} \end{bmatrix}}_{\boldsymbol{x}_{\mathcal{D}_t}} + \boldsymbol{\varepsilon}_{\mathcal{D}}, \tag{18}$$

where $x_{\mathcal{D}_t}, y_{\mathcal{D}_t}$ denote the position, and $\dot{x}_{\mathcal{D}_t}, \dot{y}_{\mathcal{D}_t}$ denote the velocity in x and y direction in the global reference frame, respectively. $\boldsymbol{\varepsilon}_{\mathcal{D}} \sim \mathcal{N}(0, \Sigma_{\mathcal{D}})$ is a random variable with Gaussian noise distribution with covariance matrix $\Sigma_{\mathcal{D}}$. In the following, it is assumed that the robot is able to detect and track a dynamic obstacle at each time instant for example with the methods in [32] and [33]. It is often sufficient to assume a linear motion of a dynamic obstacle [34]. A linear model allows a one time computation of the mean state and the covariance of the dynamic obstacle using the Kalman filter [30]

$$\boldsymbol{\mu}_{\boldsymbol{x}_{\mathcal{D}_{t+1}}} = \boldsymbol{A}^t \boldsymbol{\mu}_{\boldsymbol{x}_{\mathcal{D}_0}} \tag{19}$$

$$\Sigma_{\boldsymbol{x}_{\mathcal{D}_{t+1}}} = \boldsymbol{A}^{t+1} \Sigma_{\boldsymbol{x}_{\mathcal{D}_0}} \left(\boldsymbol{A}^{\mathrm{T}}\right)^{t+1} + \sum_{k=0}^{t} \boldsymbol{A}^{t-k-1} \Sigma_{\mathcal{D}} \left(\boldsymbol{A}^{\mathrm{T}}\right)^{t-k-1}, \tag{20}$$

where $\boldsymbol{\mu}_{\boldsymbol{x}_{\mathcal{D}_{t+1}}}$ and $\Sigma_{\boldsymbol{x}_{\mathcal{D}_{t+1}}}$ are the mean and the covariance matrix of the dynamic obstacle at time instant $t+1$, respectively. Process noise is used to increase the uncertainty while propagating the mean state and covariance in time, which increases robustness without specific knowledge of the dynamic obstacles' behavior. In this case, the process noise of the dynamic obstacle can be tuned based on confidence of the unknown future behavior of the dynamic obstacle. If it is certain that the dynamic obstacle will continue moving in the measured direction at the measured velocity, small values for the process noise matrix can be selected as parameters inside the algorithm. If the dynamic obstacle behaved in a nonlinear manner in previous time instants, larger values for the process noise should be selected.

D. PROBABILISTIC COLLISION CHECK

This section introduces two methods for calculating the probability of collision.

1) CHANCE CONSTRAINTS

The objective is to guarantee that the probability of a collision with any obstacle is less than the assigned value Γ. In a deterministic world, a collision will occur if (1) is satisfied. Hence, in belief space context the inequality, i.e.,

$$p(\boldsymbol{x}_t \in \mathcal{C}_{\text{obs}}) = p\left(\bigwedge_{i=1}^{n_e} \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{q}_t < b_{it}\right) \leq \Gamma \tag{21}$$

has to be evaluated. Here, b_{it} is only time dependent for a dynamic obstacle and a is not time dependent since we assume no rotation of obstacles. Since normality is preserved by linear transformations [35], a new random variable, $d \sim \mathcal{N}(\mu_d, \sigma_d^2)$ can be introduced with

$$\mu_d = \boldsymbol{a}^{\mathrm{T}} \boldsymbol{\mu}_{\boldsymbol{q}_t} - b \tag{22}$$

$$\sigma_d^2 = \boldsymbol{a}^{\mathrm{T}} \Sigma_{\boldsymbol{q}_t} \boldsymbol{a}, \tag{23}$$

where $\boldsymbol{\mu}_{\boldsymbol{q}_t}$ is the mean value of the position of the mobile robot with respective covariance matrix $\Sigma_{\boldsymbol{q}_t}$ at time instant t. Mean and covariance are taken from the mean vector and covariance matrix of robot state $\boldsymbol{x}_t$, respectively. Therefore, the new inequality condition for the probability of collision is as follows:

$$p(d < 0) \leq \Gamma. \tag{24}$$

This probabilistic constraint is shown to be equivalent to the following deterministic constraint using the mean and variance of d, together with the inverse Gaussian error function [17], [18]:

$$\mu_d \geq \sqrt{2\sigma_d^2}\, \mathrm{erf}^{-1}\left(1 - 2\Gamma\right). \tag{25}$$

The left-hand side of the above equation is the mean distance between the center of the robot and the line segment of the obstacle, and the right-hand side represents the minimum allowed distance between the center of the robot and the line segment. Hence, the chance constraint is satisfied if

$$\bigvee_{i=1}^{n_e} \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{\mu}_{\boldsymbol{q}_t} - b_{it} \geq \sqrt{2\boldsymbol{a}_i^{\mathrm{T}} \Sigma_{\boldsymbol{q}_t} \boldsymbol{a}_i}\, \mathrm{erf}^{-1}\left(1 - 2\Gamma\right) \tag{26}$$

is true. This approach can be extended for obstacles with an uncertain position and translation [19]. The current work assumes static and dynamic obstacles. The mean distance μ_d is unchanged for static obstacles and can be calculated for dynamic obstacles with updated parameters b_{it} based on the translation of the dynamic obstacle at time instant t. The variance for d for both static and dynamic obstacles is as follows:

$$\sigma_d^2 = \boldsymbol{a}^{\mathrm{T}} \left(\Sigma_{\boldsymbol{q}_t} + \Sigma_{\mathcal{O}}\right) \boldsymbol{a}, \tag{27}$$

where $\Sigma_{\mathcal{O}}$ is the covariance matrix of the respective obstacle. Then, the final inequality equations for a single obstacle are shown as below:

$$\bigvee_{i=1}^{n_e} \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{\mu}_{\boldsymbol{q}_t} - b_{it} \geq \sqrt{2\boldsymbol{a}_i^{\mathrm{T}} \left(\Sigma_{\boldsymbol{q}_t} + \Sigma_{\mathcal{O}}\right) \boldsymbol{a}_i}\, \mathrm{erf}^{-1}\left(1 - 2\Gamma\right). \tag{28}$$

This inequality condition has to be extended for multiple obstacles, which can be obtained using an upper bound for the probability of collision with at least one obstacle using Boole's inequality [17], i.e.,

$$p(\boldsymbol{x}_t \in \mathcal{C}_{\text{obs}}) \leq \sum_{j=1}^{n_{\mathcal{O}}} p\left(\bigwedge_{i=1}^{n_{\mathcal{O}_j}} \boldsymbol{a}_{jit}^{\mathrm{T}} \mu_{q_t} < b_{jit}\right) \leq \sum_{j=1}^{n_{\mathcal{O}}} \gamma_j = \Gamma, \tag{29}$$

where $n_{\mathcal{O}_j}$ represents the number of line-segments for obstacle j and $n_{\mathcal{O}}$ is the total number of obstacles. It is shown that this inequality can only be guaranteed if

$$\gamma_j = \gamma = \frac{\Gamma}{n_{\mathcal{O}}}. \tag{30}$$

Hence, the conjunction of inequalities at each time instant t, i.e.,

$$\bigwedge_{j=1}^{n_{\mathcal{O}}} \left(\bigvee_{i=1}^{n_{\mathcal{O}_j}} \mu_{d_{ji}} \geq \sqrt{2\sigma_{d_{ji}}^2}\, \mathrm{erf}^{-1}\left(1-2\gamma\right) \right) \tag{31}$$

has to be satisfied to guarantee safe execution. Hereafter, the HA* algorithm that employs this method for the probabilistic collision check will be referred to as the chance constraint HA* (CCHA*).

2) EXACT COLLISION PROBABILITY

The inverse Gaussian error function will be over-conservative for an increasing number of obstacles in the configuration space (see (30)). Hence, many nodes will be pruned and a solution may not be found in a many-obstacle case. This drawback can be circumvented using the Gaussian error function directly to obtain the exact probability of collision for the respective obstacle [19]:

$$\delta_{jit} = \frac{1}{2}\left[1 - \mathrm{erf}\left(\frac{\boldsymbol{a}_{ji}^{\mathrm{T}}\mu_{q_t} - b_{jit}}{\sqrt{2\boldsymbol{a}_{ji}^{\mathrm{T}}\left(\Sigma_{\boldsymbol{q}_t} + \Sigma_{\mathcal{O}}\right)\boldsymbol{a}_{ji}}}\right)\right], \tag{32}$$

where δ_{jit} is the probability of collision with line segment i of obstacle j at time instant t. The inequality equation for all obstacles at one time instant using the Boole's inequality is as follows

$$p(\boldsymbol{x}_t \in \mathcal{C}_{\mathrm{obs}}) \leq \textstyle\sum_{j=1}^{n_{\mathcal{O}}} \min_{i=1,\cdots,n_{\mathcal{O}_j}} \delta_{jit} = \Delta_t(\boldsymbol{x}_t), \tag{33}$$

where $\Delta_t(\boldsymbol{x}_t)$ is the probability of collision for $\boldsymbol{x}_t$ at time instant t. The CCHA* algorithm that uses this method are referred to as CCxHA*, where "x" indicates that the algorithm uses the Gaussian error function and as such calculates the exact probability of collision per obstacle.

E. COST FUNCTION EXTENSION

This section introduces an extension of the cost function for the HA* algorithm

$$\begin{aligned} E[J(\boldsymbol{x}_t)] = {} & \mu_{l_t}(1 + r_t C_{\mathrm{rev}}) \\ & + |r_t - r_{t-1}| C_{\mathrm{sw}} \\ & + k \ln(1 - \Delta_t(\boldsymbol{x}_t)). \end{aligned} \tag{34}$$

A new soft constraint is added for considering the probability of collision for the mobile robot, where $k \leq 0$ is a tuning gain employed as a trade-off between traveled distance and path safety. Instead of pruning nodes that do not satisfy the probability of collision Γ, the cost function penalizes the cost of nodes with high probability of collision. This approach has the advantage that no nodes will be pruned, and accordingly the algorithm may find a path where other approaches will fail. This path may, however have a high probability of collision. Hereafter, all HA* algorithms using the adapted cost function are labeled SCHA*, where SC refers to "soft constraint".

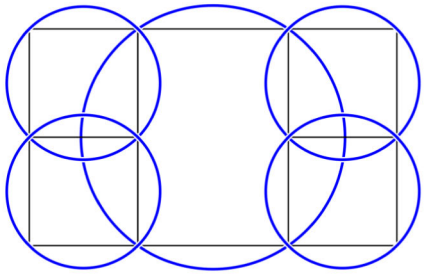


FIGURE 3. Segmentation of a rectangular robot into smaller segments.

IV. ARBITRARY-SHAPED MOBILE ROBOTS

In general, path planning algorithms work with point robots [4]. This can be achieved by inflating the obstacles based on the robot's shape and heading. For circular mobile robots, the obstacles simply have to be inflated by the radius of the robot. However, for arbitrary-shaped mobile robots, the obstacles must be inflated using the Minkowski difference [13], where each possible heading also has to be considered. This can computationally and memory-wise be very demanding. Uncertainty in the robot's heading complicates the inflation. One simplification for the configuration space is the segmentation of the robot into smaller circular parts to achieve a similar result as for circular mobile robots [28], where the obstacles will be inflated with the radius of the respective segment of the robot. In this work, the robot is separated into several smaller circles as shown in Fig. 3.

The collision check for the center of motion is the same as in (31) and (33) using the parameters for the inflated obstacle. For the other segments, the uncertainty of the heading has to be included. The general equation for deriving the center of each segment is as follows:

$$\begin{bmatrix} x_t^j \\ y_t^j \end{bmatrix} = \underbrace{\begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{bmatrix} \begin{bmatrix} l_{x_t^j} \\ l_{y_t^j} \end{bmatrix}}_{g(\boldsymbol{x}_t)}, \tag{35}$$

where (x_t^j, y_t^j) is the position of the center of a selected square segment j at time step t, and indicates the selected square segment, and $(l_{x_t^j}, l_{y_t^j})$ is the corresponding translation value. This transformation is nonlinear, which means that resulting variables are no longer Gaussian [35]. Therefore, the transformation will be approximated by the Taylor series expansion as follows:

$$\begin{aligned} \begin{bmatrix} x_t^j \\ y_t^j \end{bmatrix} & \approx g(\boldsymbol{\mu}_{\boldsymbol{x}_t}) + \nabla\, g(\boldsymbol{x}_t)|_{\boldsymbol{\mu}_{\boldsymbol{x}_t}} \left(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}_t}\right) \\ & = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} -\mathrm{s}_{\mu_{\theta_t}} & -\mathrm{c}_{\mu_{\theta_t}} \\ \mathrm{c}_{\mu_{\theta_t}} & -\mathrm{s}_{\mu_{\theta_t}} \end{bmatrix} \begin{bmatrix} l_{x_t^j} \\ l_{y_t^j} \end{bmatrix} \theta_t \\ & \quad + \begin{bmatrix} \mathrm{c}_{\mu_{\theta_t}} + \mathrm{s}_{\mu_{\theta_t}}\mu_{\theta_t} & -\mathrm{s}_{\mu_{\theta_t}} + \mathrm{c}_{\mu_{\theta_t}}\mu_{\theta_t} \\ \mathrm{s}_{\mu_{\theta_t}} - \mathrm{c}_{\mu_{\theta_t}}\mu_{\theta_t} & \mathrm{c}_{\mu_{\theta_t}} + \mathrm{s}_{\mu_{\theta_t}}\mu_{\theta_t} \end{bmatrix} \begin{bmatrix} l_{x_t^j} \\ l_{y_t^j} \end{bmatrix}, \end{aligned} \tag{36}$$

where $\mathrm{s}_{\mu_{\theta_t}}$ and $\mathrm{c}_{\mu_{\theta_t}}$ are abbreviations for $\sin\mu_{\theta_t}$ and $\cos\mu_{\theta_t}$, respectively; x_t^j and y_t^j on the right-hand side of this equation

are of the form $x_t^j = a_1 x_t + b_1 \theta_t + c_1$ and $y_t^j = a_2 y_t + b_2 \theta_t + c_2$ with the constants $a_1, a_2, b_1, b_2, c_1, c_2$. Hence, it is sufficient to calculate the expectation and variance for the general case, i.e., $\mathrm{Z} = a\mathrm{X} + b\mathrm{Y} + c$, where a, b, and c are constants and X, Y, and Z are random variables. Subsequently, the expectation and variance are calculated in terms of moment expression. Since it is a linear transformation, the expectation for Z is as follows:

$$E[\mathrm{Z}] = E[a\mathrm{X} + b\mathrm{Y} + c] = aE[\mathrm{X}] + bE[\mathrm{Y}] + c. \quad (37)$$

The complete calculation of the variance is shown in Appendix A, and the resulting equation is as follows:

$$\mathrm{var}(\mathrm{Z}) = a^2 \mathrm{var}(\mathrm{X}) + b^2 \mathrm{var}(\mathrm{Y}) + 2ab\,\mathrm{cov}(\mathrm{X}, \mathrm{Y}). \quad (38)$$

Therefore, the mean and variance for the transformed coordinates are

$$\begin{bmatrix} \mu_{x_t^j} \\ \mu_{y_t^j} \end{bmatrix} = \begin{bmatrix} \mu_{x_t} \\ \mu_{y_t} \end{bmatrix} + \begin{bmatrix} \mathrm{c}_{\mu_{\theta_t}} & -\mathrm{s}_{\mu_{\theta_t}} \\ \mathrm{s}_{\mu_{\theta_t}} & \mathrm{c}_{\mu_{\theta_t}} \end{bmatrix} \begin{bmatrix} l_{x_t^j} \\ l_{y_t^j} \end{bmatrix} \quad (39)$$

$$\begin{aligned} \begin{bmatrix} \sigma^2_{x_t^j} \\ \sigma^2_{y_t^j} \end{bmatrix} &= \begin{bmatrix} \sigma^2_{x_t} \\ \sigma^2_{y_t} \end{bmatrix} + \begin{bmatrix} \mathrm{s}^2_{\mu_{\theta_t}} l^2_{x_t^j} + \mathrm{s}_{\mu_{\theta_t}} \mathrm{c}_{\mu_{\theta_t}} l_{x_t^j} l_{y_t^j} + \mathrm{c}^2_{\mu_{\theta_t}} l^2_{y_t^j} \\ \mathrm{c}^2_{\mu_{\theta_t}} l^2_{x_t^j} - \mathrm{s}_{\mu_{\theta_t}} \mathrm{c}_{\mu_{\theta_t}} l_{x_t^j} l_{y_t^j} + \mathrm{s}^2_{\mu_{\theta_t}} l^2_{y_t^j} \end{bmatrix} \sigma^2_{\theta_t} \\ &\quad + 2 \begin{bmatrix} \sigma_{x_t,\theta_t} & 0 \\ 0 & \sigma_{y_t,\theta_t} \end{bmatrix} \begin{bmatrix} -\mathrm{s}_{\mu_{\theta_t}} & -\mathrm{c}_{\mu_{\theta_t}} \\ \mathrm{c}_{\mu_{\theta_t}} & -\mathrm{s}_{\mu_{\theta_t}} \end{bmatrix} \begin{bmatrix} l_{x_t^j} \\ l_{y_t^j} \end{bmatrix} \end{aligned} \quad (40)$$

Then, $\mathrm{cov}(x_t^j, y_t^j)$ can be calculated in a similar manner (the full equation is shown in Appendix A):

$$\begin{aligned} &\mathrm{cov}(a_1 x_t + b_1 \theta_t + c_1, a_2 y_t + b_2 \theta_t + c_2) \\ &\quad = a_1 a_2 \mathrm{cov}(x_t, y_t) + a_1 b_2 \mathrm{cov}(x_t, \theta_t) \\ &\quad\quad + a_2 b_1 \mathrm{cov}(y_t, \theta_t) + b_1 b_2 \mathrm{var}(\theta_t). \end{aligned} \quad (41)$$

The covariance can be expressed in vector form as follows:

$$\begin{aligned} \sigma_{x_t^j, y_t^j} &= \sigma_{x_t, y_t} + \begin{bmatrix} \sigma_{x_t,\theta_t} \\ \sigma_{y_t,\theta_t} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -\mathrm{s}_{\mu_{\theta_t}} & -\mathrm{c}_{\mu_{\theta_t}} \\ \mathrm{c}_{\mu_{\theta_t}} & -\mathrm{s}_{\mu_{\theta_t}} \end{bmatrix} \begin{bmatrix} l_{x_t^j} \\ l_{y_t^j} \end{bmatrix} \\ &\quad + \left[\mathrm{s}_{\mu_{\theta_t}} \mathrm{c}_{\mu_{\theta_t}} \left(l^2_{y_t^j} - l^2_{x_t^j} \right) - \mathrm{c}_{2\mu_{\theta_t}} l_{x_t^j} l_{y_t^j} \right] \sigma^2_{\theta_t}. \end{aligned} \quad (42)$$

Hence, the covariance matrix for the square center positions is

$$\Sigma_{\boldsymbol{q}_j} = \begin{bmatrix} \sigma^2_{x_t^j} & \sigma_{x_t^j, y_t^j} \\ \sigma_{x_t^j, y_t^j} & \sigma^2_{y_t^j} \end{bmatrix}. \quad (43)$$

The above equations can be used to calculate the probability of collision for each of the square centers with either (31) or (33) using the inflated obstacle parameters, where the maximum probability of collision of all segment centers will be used per obstacle. An example figure of the resulting confidence ellipses for the square segments is shown in Fig. 4, where the robot's heading is 45° and the selected covariance matrix of the robot is

$$\Sigma_{\boldsymbol{x}} = \begin{bmatrix} 0.003 & -0.0009 & -0.00005 \\ -0.0009 & 0.003 & -0.00005 \\ -0.00005 & -0.00005 & 0.05 \end{bmatrix}.$$

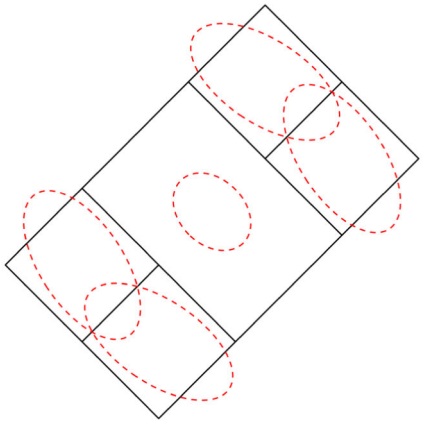


FIGURE 4. Rectangular robot with 95 % confidence ellipses.

It is shown that the confidence ellipses for four square centers are angled and have larger radii than the ellipse of the robot's center. This follows from the fact that variance σ^2_θ of the heading and covariances $\sigma_{x,\theta}, \sigma_{y,\theta}$ are considered. Hereafter, algorithms (i.e., HA*, CCHA*, CCxHA*, and SCHA*) that incorporate the shape of the robot are labeled with the prefix ASR (arbitrary-shaped robot).

V. RESULTS

This section demonstrates the validity of the approaches applied in this paper. The variants of the proposed algorithms (i.e., CCHA*, CCxHA*, SCHA*, ASR CCHA*, ASR CCxHA* and ASR SCHA*) are examined to evaluate their strengths and weaknesses. The above proposed algorithms are compared with existing approaches in a clustered, static environment, and an environment with a dynamic obstacle. For the comparison, the conventional A* [6], the HA* algorithm [2], [4], ASR HA* considering the shape of the robot using the approach of [28], the RRT algorithm [12], the closed-loop RRT (CLRRT) algorithm [1], [36], the heuristic arrival time field-biased random tree (HeAT-RT) [14] well as the probabilistic approaches CCRRT and online CCRRT (CCxRRT) [19], [24], which are based on CLRRT, are all implemented. All kinematic planners use the same motion model, and all probabilistic planners use the same uncertainties and uncertainty propagation method. The RRT-based planners for comparison are random in their search, which generate different solution paths in every calculation, and therefore each RRT algorithm is executed 1000 times for both environments. The success rate, shortest distance, mean computation time, standard deviation σ_{time}, mean distance, and standard deviation of the distance σ_{dist} are calculated for evaluation. A plan is considered successful if it finds a solution within 20 s. Furthermore, RRT-based planners are generally unable to exactly reach the goal pose, but the vicinity of $\boldsymbol{\mu}_{\mathrm{goal}} \in \mathcal{X}_{\mathrm{goal}} \subset \mathrm{SE}(2)$. Here, the vicinity to the goal is defined to be in a range of 0.5 m with a heading difference of less than 20°.

An evaluation of the proposed cost function (see (34)) is presented. Finally, the performance of the algorithms is evaluated using MCS [6].

The robot used for simulation has a length of 1.27 m and a width of 0.75 m. The robot will move with a fixed linear velocity of $v = 0.5\,\mathrm{m/s}$ (except for HeAT-RT) and, for the HA*-based planners, an angular velocity of $\omega = 10\pi/180$ rad/s. Furthermore, the environments are discretized with a cell size of 0.5 m and an angle increment of $5\pi/180$ rad. The penalties for driving in reverse and switching the driving direction are set to the same values, i.e., $C_{\mathrm{rev}} = 1.0$ and $C_{\mathrm{sw}} = 1.0$, respectively. The algorithms are tested in ROS Kinetic [37] on a GNU/Linux Ubuntu 16.04 laptop with an Intel® Core™i7-8550U CPU @ 1.80 GHz × 8 and 8 GB memory.

A. RESULTS ON STATIC ENVIRONMENT

In this section, the algorithms are compared in a static environment with nine obstacles (see Fig. 5). The process noise and initial covariance matrix for the robot are as follows:

$$\Sigma_M = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.0005 \end{bmatrix}$$
$$\Sigma_{\boldsymbol{x}_0} = 0.0001 I_3.$$

The covariance matrix for the static obstacles is

$$\Sigma_{\mathcal{S}} = 0.1 I_2.$$

The gain for the soft constraint is $k = -1.5$ and the chance constraint parameter is $\Gamma = 0.25$.

Fig. 5 shows the resulting path attaining the shortest distance (solid red) and expanded nodes (solid orange) for the ASR CCxHA* algorithm. It is shown that the algorithm pruned all nodes that were too close to the obstacles, and the final path attempted to keep as much distance as possible from all obstacles. The resulting path attaining the shortest distance (solid red) using the CCRRT algorithm and a path with ''zig-zag'' motions (dash-dotted blue) using the CCxRRT algorithm are shown in Fig. 6. The path segments with ''zig-zag'' motions are emphasized with black circles. The paths for all algorithms are shown in Fig. 7. Both the A* and RRT algorithms move close to obstacles to generate short distance paths. The conventional HA* algorithm, which did not consider the robot shape, moved close to the obstacles while CCHA* moved very conservatively, although it did not consider the shape of the robot. The ASR CCHA* was not able to find a solution (not shown in the figure) because it pruned all nodes that were considered ''too dangerous''. This result was derived from the number of obstacles, where the allowed probability of collision per obstacle (see (30)), i.e., $\gamma \approx 0.028$ was very conservative in terms of satisfying Boole's inequality. The CCxHA* and ASR CCxHA* algorithms were able to find a solution that guaranteed $\Delta_t(\boldsymbol{x}_t) \leq \Gamma$ for $\forall t$. Furthermore, the SCHA* and ASR SCHA* algorithms found robust solutions without pruning any nodes. The HA* algorithms move the robot in reverse only if the forward motion violates the collision constraints, whereas the CLRRT-based algorithms take a backward motion based on the position of the randomly expanded node. Therefore, the best solution of the CLRRT-based algorithms took a different route than the other algorithms. Table 1 shows the performance of the algorithms. The probabilistic approaches required more time to check for collisions than the conventional HA* and CLRRT algorithms. CCHA* and CCRRT use the same method for collision check and have, therefore, the same collision check computation time per node. Accordingly, the collision check time per node is the same for CCxHA* and CCxRRT. The CCHA* algorithm, which used the inverse Gaussian error function, required approximately 10 times more time per node than the HA* algorithm, while the algorithms that employed a Gaussian error function (CCxHA*, ASR CCxHA*, SCHA* and ASR SCHA*) needed slightly more; this was because they calculated the Gaussian error function for each line segment. Additional time per node was needed if the shape of the robot was considered. It is shown that a trade-off occurred between computation time and robustness. The RRT-based planners were able to find paths that are shorter in distance than the HA*-based algorithms, but the mean distance of all trials was larger than the distance of any HA*-based planner. Furthermore, only 9.5 % of the paths of CCRRT were successful in finding a path within the given time frame. The mean computation time of CCRRT and CCxRRT is much larger than that of CCHA* and CCxHA*. In addition, the standard deviation of the distance and computation time is large for the CCRRT and CCxRRT path planners. The HeAT-RT algorithm expanded fewer nodes than most other planners but generated a path near an obstacle and had a larger mean computation time compared to the proposed probabilistic algorithm CCxHA*. In addition, the HeAT-RT had the largest standard deviation of the distance.

The best performance of the proposed algorithms in the static environment in terms of travelling distance and computation time are provided by the CCxHA* and the ASR SCHA*.

B. RESULTS IN A DYNAMIC ENVIRONMENT

In this section, the algorithms are evaluated in a dynamic environment. The dynamic obstacle had a square size of 0.5 m with linear velocity $v = 0.3536\,\mathrm{m/s}$ and a heading direction of 135°. The initial covariance matrix and process noise of the dynamic obstacle are given as follows:

$$\Sigma_{\boldsymbol{x}_{\mathcal{D}_0}} = 0.001\, I_4$$
$$\Sigma_{\mathcal{D}} = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0.0001 \end{bmatrix}.$$

The covariance matrix of the static obstacles was the same as in the static environment, and elements of the covariance matrix and the process noise of the mobile robot were increased for this environment, where $\Sigma_{\boldsymbol{x}_0} = 0.01 I_3$ and the

TABLE 1. Traveled distance and computation time per node in static environment.

Algorithm	Shortest distance [m]	Collision check time per node [µs]	Expanded nodes (best solution)	Mean comp. time [s]	σ_{time} [s]	Mean distance [m]	σ_{dist} [m]	Successful nodes [%]
CCHA*	23.70	~ 149	2620	~ 1.21	—	—	—	—
CCxHA*	20.51	~ 240	408	~ 0.22	—	—	—	—
SCHA*	20.98	~ 222	1333	~ 0.592	—	—	—	—
ASR CCHA*	x	x	x	x	—	—	—	—
ASR CCxHA*	22.83	~ 898	5086	~ 13.84	—	—	—	—
ASR SCHA*	22.15	~ 885	2022	~ 3.5	—	—	—	—
A*	22.63	~ 15	509	~ 0.019	—	—	—	—
HA*	20.19	~ 15	334	~ 0.02	—	—	—	—
ASR HA*	21.40	~ 67	580	~ 0.11	—	—	—	—
RRT	17.70	~ 15	191	~ 0.022	~ 0.02	24.04	3.49	100
CLRRT	18.64	~ 15	1720	~ 0.57	~ 0.74	28.25	4.82	100
CCRRT	20.47	~ 149	10628	~ 8.40	~ 5.64	30.78	4.32	9.5
CCxRRT	19.25	~ 240	10242	~ 2.65	~ 3.02	28.39	4.71	97.5
HeAT-RT	18.90	~ 15	288	~ 0.32	~ 0.51	26.51	5.39	100

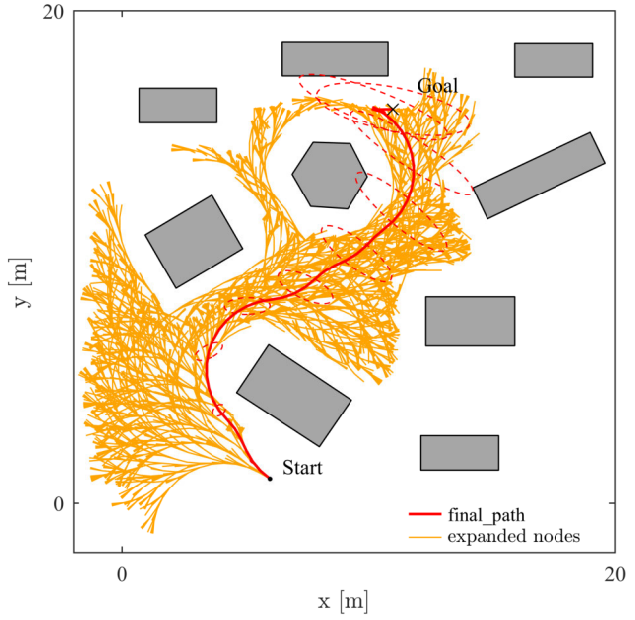


FIGURE 5. The resulting path (solid red) and all expanded nodes (solid orange) using the ASR CCxHA* algorithm (the dashed ellipses are plotted every 5th node with a confidence of 95 %).

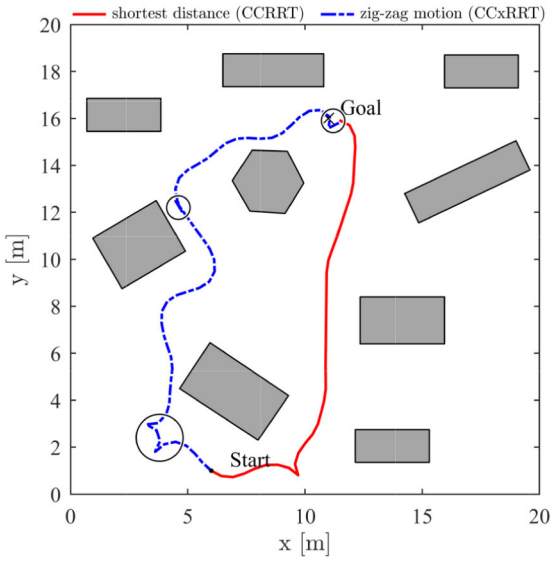


FIGURE 6. The resulting path attaining the shortest distance (solid red) using the CCRRT algorithm and a "zig-zag" motion (dash-dotted blue) using the CCxRRT algorithm (the black circles emphasize the path segments with a "zig-zag" motion).

process noise is given as follows:

$$\Sigma_M = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.005 \end{bmatrix}.$$

The parameters for the chance constraints and the soft constraint are $\Gamma = 0.4$ and $k = -1.5$, respectively.

Results are shown in Figures 8 and 9, where the former shows the result of the ASR CCHA* algorithm at a selected node. The robot attempted to cross the path of the dynamic obstacle behind it while considering its own uncertainty and that of the dynamic obstacle. Fig. 9 shows the results for all the algorithms. Most HA* algorithm variants crossed the path of the dynamic obstacle behind it, except for the HA* algorithm, the CCxHA* algorithm, which did not consider the robot shape and the ASR SCHA* algorithm, which conservatively crossed the path of the dynamic obstacle ahead of it. Table 2 shows the results for each of the planner. It is shown that the SCHA* variations expanded many nodes and therefore had a larger computation time. Due to the dynamic obstacle, the soft constraint part increased, which will lead to the expansion of many nodes. Furthermore, most of the algorithms had a similar traveled distance as the HA* and ASR HA* algorithms. The CCxRRT algorithm had a similar computation time as the CCxHA*, but a larger mean distance with a large standard deviation. The CCRRT algorithm had the second-largest mean computation time, which is more than two times larger as for the ASR CCHA*. Both the A* and RRT moved, as in the static environment, very close to the obstacle. Furthermore, the HeAT-RT expanded the fewest nodes but had the largest standard deviation of the

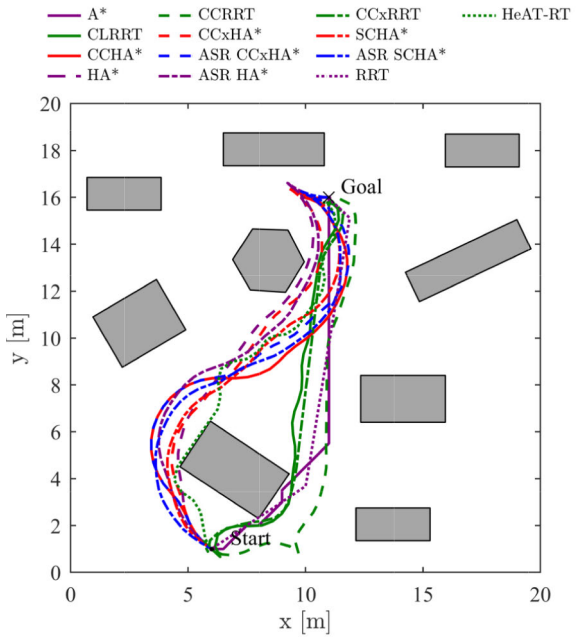


FIGURE 7. Resulting paths for all algorithms in a clustered, static environment.

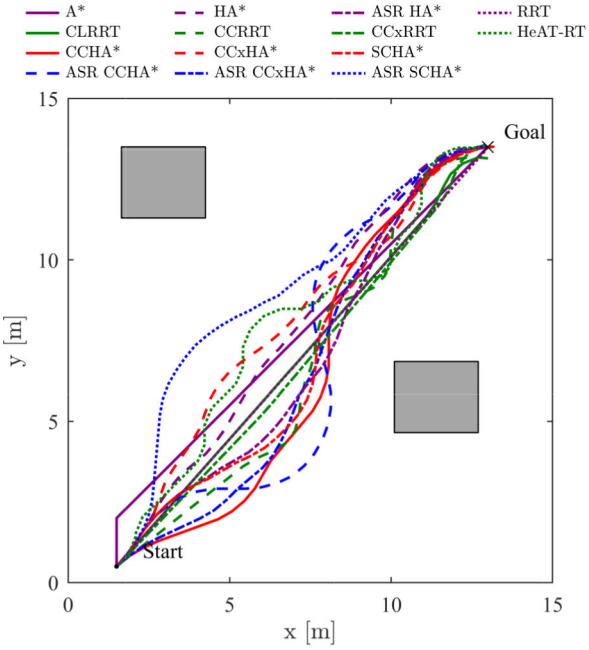


FIGURE 9. Resulting paths in the dynamic environment.

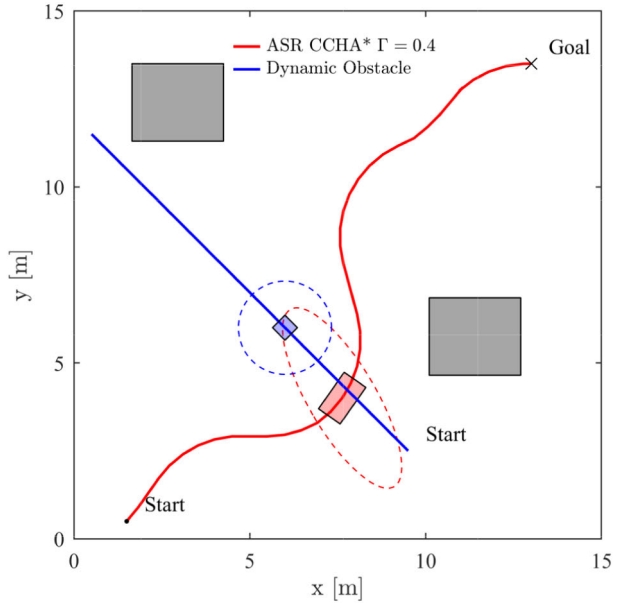


FIGURE 8. The ASR CCHA* algorithm result at a selected node.

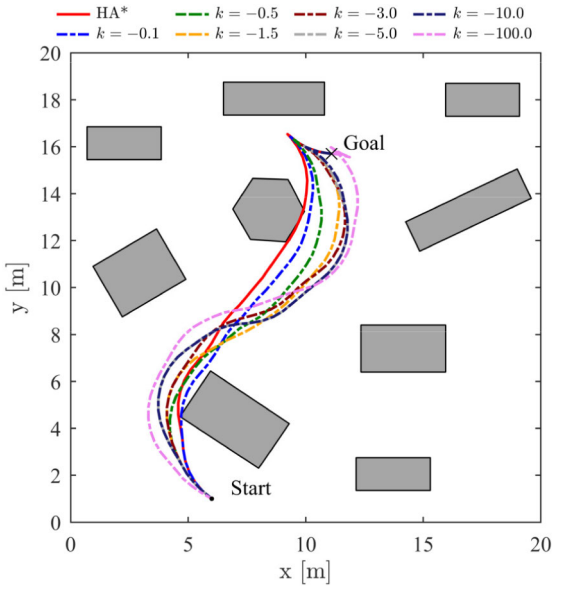


FIGURE 10. Results for different gains of the SCHA* algorithm.

distance and a larger mean computation time compared to the proposed CCHA*, CCxHA* and SCHA* algorithms.

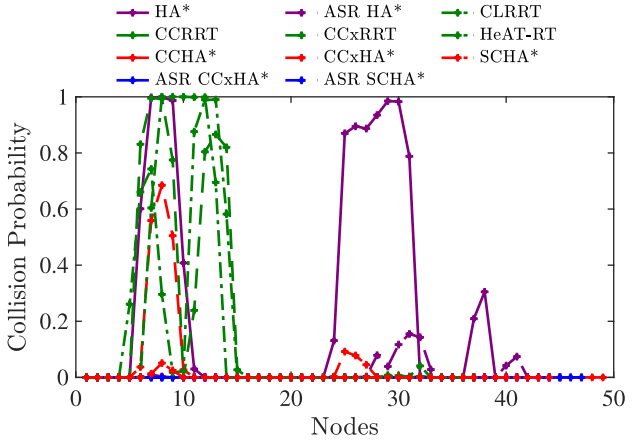
The CCxHA* and ASR CCxHA* performed best among the proposed algorithms in terms of computation time and travelling distance.

C. RESULTS FOR DIFFERENT SOFT CONSTRAINT GAINS

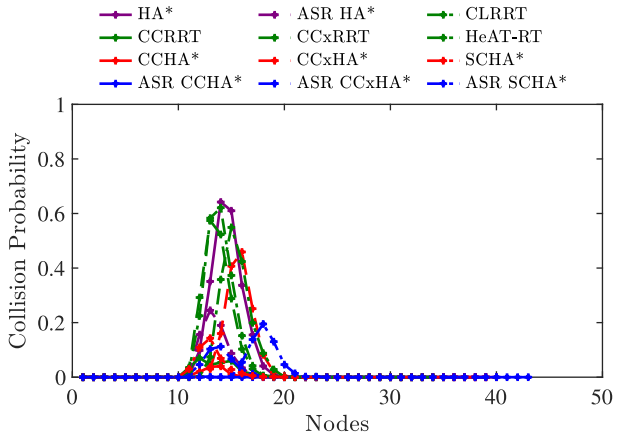
This section compares the result of the HA* algorithm with those of the SCHA* algorithm using different k gains for the soft constraint. Fig. 10 shows the resulting paths, where the conservatism of the SCHA* algorithm increases with a decreasing gain value for k. Furthermore, Table 3 shows that the number of expanded nodes, the traveled distance and the computation time also increases. For $k = -0.1$, the traveled distance is less than for the HA* algorithm (see Table 1); however the SCHA* algorithm with $k = -0.1$ moves close to the obstacles. The traveled distance of the SCHA* algorithm with $k = -0.5$ and $k = -1.5$ is similar to the traveled distance of HA*. Based on these results, the gain value of the soft constraint should be set between -0.5 and -1.5

TABLE 2. The traveled distance and computation time per node in a dynamic environment.

Algorithm	Shortest distance [m]	Collision check time per node [µs]	Expanded nodes (best solution)	Mean comp. time [s]	σ_{time} [s]	Mean distance [m]	σ_{dist} [m]	Successful nodes [%]
CCHA*	18.75	∼ 179	745	∼ 0.39	—	—	—	—
CCxHA*	18.25	∼ 224	550	∼ 0.3	—	—	—	—
SCHA*	18.75	∼ 212	2064	∼ 0.9	—	—	—	—
ASR CCHA*	20.35	∼ 410	1063	∼ 1.32	—	—	—	—
ASR CCxHA*	18.36	∼ 637	671	∼ 1.08	—	—	—	—
ASR SCHA*	18.84	∼ 589	4138	∼ 4.9	—	—	—	—
A*	18.38	∼ 14	447	∼ 0.011	—	—	—	—
HA*	17.97	∼ 14	309	∼ 0.02	—	—	—	—
ASR HA*	18.25	∼ 56	584	∼ 0.11	—	—	—	—
RRT	17.36	∼ 14	207	∼ 0.016	∼ 0.005	20.28	2.33	100
CLRRT	17.46	∼ 14	699	∼ 0.27	∼ 0.57	20.31	2.68	100
CCRRT	18.85	∼ 179	4653	∼ 2.81	∼ 3.65	26.19	4.44	93.9
CCxRRT	17.60	∼ 224	788	∼ 0.55	∼ 0.71	21.79	3.52	100
HeAT-RT	19.05	∼ 14	72	∼ 0.96	∼ 0.51	25.53	4.73	99.9



(a) Collision probability for the static environment.



(b) Collision probability for the dynamic environment.

FIGURE 11. Monte Carlo simulation for static and dynamic environments.

to achieve a good traveling distance, computation time, and safety.

D. PERFORMANCE COMPARISON

This work adapted the MCS algorithm in [38], which was modified for dynamic obstacles and robots with nonlinear dynamics, where the robot state was updated at each expanded node for the motion command of the final path. Algorithm 1 shows a pseudo-code of the MCS algorithm. Variable $\overline{p_{\text{coll}}(\mathcal{R},\cdot)}$ denotes the probabilities for robot $\mathcal{R}$ to not collide with the tested obstacle. $\mathcal{S}^i, \mathcal{D}^j$ denote static obstacle i and dynamic obstacle j, respectively. $\mathcal{A}_{\mathcal{R}}$, $\mathcal{A}_{\mathcal{S}^i}$ and $\mathcal{A}_{\mathcal{D}^j}$ denote the area of the robot $\mathcal{R}$, static obstacle i and dynamic obstacle j, respectively. $\mathcal{O}$ is the set of all obstacles and $n_{\mathcal{S}}, n_{\mathcal{D}}$ is the number of static and dynamic obstacles, respectively. Function *randc*() calculates the Gaussian random variable. This work used $P = 1000$ particles for the MCS, and the probability of collision was calculated based on the shape of the robot. The results of the static and dynamic environments are shown in Fig. 11. The red lines indicate the MCS results for the proposed algorithms assuming a point robot, the blue lines indicate the algorithms that consider the shape of the robot, and the green and purple lines show the results of the planners used for comparison, except A* and RRT, which cannot be evaluated using the MCS since they do not use a robot model. The probability of collision was significantly reduced for the static environment. Only CCxRRT and the proposed CCxHA* had high values, likely because they did not consider the shape of the robot and allowed a probability of collision of $\Delta_t(\boldsymbol{x}_t) \leq 0.25$. The result of the dynamic environment was similar. In both environments, the CCHA* algorithm variation indicated almost no probability of collision while the CCxHA* algorithm had a small value based on the setting of Γ. It can be seen that the algorithms that do not consider uncertainties had the largest probability of collision. Furthermore, the ASR variations (blue lines) of the proposed algorithms have a very low probability of collision in both the static and dynamic environments.

VI. DISCUSSION

The performance of the algorithms was evaluated using a clustered, static environment, and an environment with a dynamic obstacle. In addition, the true probability of a collision was calculated using the MCS.

Algorithm 1 Monte Carlo Simulation

procedure ProbabilisticCollisionCheck
  for all Nodes **do**
    $reset(p_{coll})$
    for $j \leftarrow n_{\mathcal{S}}$ **do**
      for $i \leftarrow P$ **do**
        $\hat{\boldsymbol{u}} \leftarrow \boldsymbol{u} + randc(0, \Sigma_M)$
        $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \hat{\boldsymbol{u}})$
        $\boldsymbol{x}_{\mathcal{S}^j} = \boldsymbol{x}_{\mathcal{S}_0^j} + randc(0, \Sigma_{\mathcal{S}})$
        $\mathcal{S}^j \leftarrow \boldsymbol{x}_{\mathcal{S}^j}$
        if $\mathcal{A}_{\mathcal{R}} \cap \mathcal{A}_{\mathcal{S}^j} = \emptyset$ **then**
          $\overline{p_{\text{coll}}(\mathcal{R}, \mathcal{S}^j)} \leftarrow \overline{p_{\text{coll}}(\mathcal{R}, \mathcal{S}^j)} + 1$
        end if
      end for
    end for
    for $j \leftarrow n_{\mathcal{D}}$ **do**
      for $i \leftarrow P$ **do**
        $\hat{\boldsymbol{u}} \leftarrow \boldsymbol{u} + randc(0, \Sigma_M)$
        $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \hat{\boldsymbol{u}})$
        $\boldsymbol{x}_{\mathcal{D}^j_{t+1}} = \boldsymbol{\mu}_{\boldsymbol{x}_{\mathcal{D}^j_{t+1}}} + randc(0, \Sigma_{\mathcal{D}^j})$
        $\mathcal{D}^j \leftarrow \boldsymbol{x}_{\mathcal{D}^j_{t+1}}$
        if $\mathcal{A}_{\mathcal{R}} \cap \mathcal{A}_{\mathcal{D}^j} = \emptyset$ **then**
          $\overline{p_{\text{coll}}(\mathcal{R}, \mathcal{D}^j)} \leftarrow \overline{p_{\text{coll}}(\mathcal{R}, \mathcal{D}^j)} + 1$
        end if
      end for
    end for
    $\overline{p_{\text{coll}}(\mathcal{R}, \mathcal{O})} \leftarrow 1$
    for $k \leftarrow 1$ to $n_{\mathcal{S}}$ **do**
      $\overline{p_{\text{coll}}(\mathcal{R}, \mathcal{O})} \leftarrow \overline{p_{\text{coll}}(\mathcal{R}, \mathcal{O})} \cdot \overline{p_{\text{coll}}(\mathcal{R}, \mathcal{S}^k)}$
    end for
    for $k \leftarrow 1$ to $n_{\mathcal{D}}$ **do**
      $\overline{p_{\text{coll}}(\mathcal{R}, \mathcal{O})} \leftarrow \overline{p_{\text{coll}}(\mathcal{R}, \mathcal{O})} \cdot \overline{p_{\text{coll}}(\mathcal{R}, \mathcal{D}^k)}$
    end for
    $p_{\text{coll}}(\mathcal{R}, \mathcal{O}) \leftarrow 1 - \frac{\overline{p_{\text{coll}}(\mathcal{R}, \mathcal{O})}}{P^n \mathcal{O}}$
    $store(p_{\text{coll}}(\mathcal{R}, \mathcal{O}))$
  end for
end procedure

The results showed that the proposed algorithms were able to find probabilistically safe paths within the set threshold, i.e., Γ. Furthermore, the CCHA* algorithm was very conservative, thus exhibited a very low probability of collision (Fig. 11). However, with an increase in the number of obstacles, the CCHA* algorithm may fail to find a solution due to the pruning of all expanded nodes. The CCxHA* algorithm showed good results with a small increase in the computation time per node compared with the CCHA* algorithm (Tables 1 and 2). All proposed algorithms were able to avoid the dynamic obstacle with a decreased probability of collision (Table 2). The method considering arbitrary-shaped mobile robots further increased probabilistic safety at the cost of additional computation time per node. Therefore, the ASR variations of CCHA*, CCxHA*, and SCHA* can be used as global approaches for predicting an initial probabilistically safe path.

The proposed soft constraint approach successfully found probabilistic robust paths for both environments. However, computation time and number of expanded nodes was large for the dynamic environment, which indicates that the heuristics approach should be extended to consider dynamic obstacles. The ASR SCHA* outperformed both ASR CCHA* and ASR CCxHA* in the static environment. The ASR CCHA* failed to find a solution and the ASR CCxHA* had a large computation time (Table 1). Moreover, results indicated that the gain values for the SCHA* algorithm should be in the range of $k = -0.5$ to -1.5 to achieve good results for safety and traveled distance (Table 3).

TABLE 3. Traveled distance and computation time per node in a static environment for the SCHA* algorithm.

Soft constraint k	Distance [m]	Expanded nodes	Comp. time [s]
-0.1	19.82	400	~ 0.174
-0.5	20.62	813	~ 0.406
-1.5	20.81	1333	~ 0.592
-3.0	21.34	1846	~ 0.936
-5.0	22.01	1713	~ 0.794
-10.0	21.88	1917	~ 0.994
-100.0	23.10	5499	~ 3.176

CCHA* and CCxHA* outperformed CCRRT and CCxRRT in terms of mean computation time and mean traveled distance (Tables 1 and 2). In addition, the results of the HA* algorithms are deterministic, whereas the results of the RRT-based planners are random. The shorter computation time of the proposed HA*-based planners is the result of two factors. First, the RRT-based planners compute a new motion command for each node, whereas the HA*-based planners use a fixed motion command that can be stored beforehand in a lookup table. Second, HA*-based planners store each expanded node in a list, and the node with the lowest total cost (heuristic cost plus cost-so-far) is selected for expansion. Such a procedure requires a heap algorithm; here, binary heap [39] is used, which has an average time complexity for inserting a value of $\mathcal{O}(\log n)$ and extracting the minimum value of $\mathcal{O}(\log n)$, where n is the number of nodes in the list. Meanwhile, the RRT-based approaches use a list for all expanded nodes, where the cost of each node has to be updated based on the newly expanded node, which requires $\mathcal{O}(n)$ time. In CLRRT, CCRRT and CCxRRT the updated list has to be sorted, which takes $\mathcal{O}(n \log n)$ time [40].

The larger mean distance of the RRT-based planners is a result of detours and "zig-zag" motions (Fig. 6) caused by random selection of new nodes in the configuration space [24]. In contrast the proposed algorithms are guided by heuristics, therefore expand nodes close to the goal, and change motion directions only if a collision constraint is violated, hence avoiding "zig-zag" motions (Fig. 5).

The heuristics approach used for the proposed algorithms do not consider dynamic obstacles, which result in a large

number of node expansions. Therefore, future studies should extend the algorithms with heuristics considering dynamic obstacles to reduce the number of expanded nodes and computation time.

VII. CONCLUSION

This work proposes novel variations of the HA* algorithm, that consider the probability of collision with an obstacle. Furthermore, a method is proposed that considers the shape of the robot in the probabilistic collision calculations. The proposed algorithms outperformed the RRT-based algorithms in terms of mean computation time and mean traveled distance. The proposed algorithms that consider the arbitrary-shaped robots further increased the probabilistic safety of the obtained paths.

Future work will modify the heuristics for dynamic obstacles and heuristics in belief space will be incorporated to reduce the number of expanded nodes of the proposed algorithms. The probabilistic robust approach will be extended for non-Gaussian uncertainties. In addition, uncertain hybrid discrete-continuous systems known as jump Markov linear systems [41], [42] will be considered.

APPENDIX A
LINEAR TRANSFORMATION OF VARIANCE AND COVARIANCE

It is known that variance and covariance can be expressed using the expectation of random variables as follows:

$$\mathrm{var}(\mathrm{X}) = E[(\mathrm{X} - E[\mathrm{X}])^2], \tag{44}$$

$$\mathrm{cov}(\mathrm{X}, \mathrm{Y}) = E[(\mathrm{X} - E[\mathrm{X}])(\mathrm{Y} - E[\mathrm{Y}])]. \tag{45}$$

Furthermore, the expectation of $a\mathrm{X} + b\mathrm{Y} + c$ is

$$E[a\mathrm{X} + b\mathrm{Y} + c] = aE[\mathrm{X}] + bE[\mathrm{Y}] + c. \tag{46}$$

Using (44) and (46)

$$\begin{aligned}&\mathrm{var}(a\mathrm{X} + b\mathrm{Y} + c)\\ &= E[(a\mathrm{X} + b\mathrm{Y} + c - E[a\mathrm{X} + b\mathrm{Y} + c])^2]\\ &= E[(a(\mathrm{X} - E[\mathrm{X}]) + b(\mathrm{Y} - E[\mathrm{Y}]) + c - c)^2]\end{aligned} \tag{47}$$

Using the principles of (46) again and the binomial formula,

$$\begin{aligned}&= E[a^2(\mathrm{X} - E[\mathrm{X}])^2] + E[b^2(\mathrm{Y} - E[\mathrm{Y}])^2]\\ &\quad + E[2ab(\mathrm{X} - E[\mathrm{X}])(\mathrm{Y} - E[\mathrm{Y}])]\end{aligned} \tag{48}$$

Hence,

$$\begin{aligned}\mathrm{var}(a\mathrm{X} + b\mathrm{Y} + c) &= a^2\mathrm{var}(\mathrm{X}) + b^2\mathrm{var}(\mathrm{Y})\\ &\quad + 2ab\,\mathrm{cov}(\mathrm{X}, \mathrm{Y}).\end{aligned} \tag{49}$$

The covariance for $\mathrm{Z}_1 = a_1\mathrm{X}_1 + b_1\mathrm{Y}_1 + c_1$ and $\mathrm{Z}_2 = a_2\mathrm{X}_2 + b_2\mathrm{Y}_2 + c_2$ can be calculated similarly, i.e.,

$$\begin{aligned}&\mathrm{cov}(\mathrm{Z}_1, \mathrm{Z}_2)\\ &= E[(a_1\mathrm{X}_1 + b_1\mathrm{Y}_1 + c_1 - E[a_1\mathrm{X}_1 + b_1\mathrm{Y}_1 + c_1])\\ &\qquad (a_2\mathrm{X}_2 + b_2\mathrm{Y}_2 + c_2 - E[a_2\mathrm{X}_2 + b_2\mathrm{Y}_2 + c_2])],\end{aligned} \tag{50}$$

which can be simplified using (46),

$$\begin{aligned}\mathrm{cov}(\mathrm{Z}_1, \mathrm{Z}_2) &= E[(a_1(\mathrm{X}_1 - E[\mathrm{X}_1]) + b_1(\mathrm{Y}_1 - E[\mathrm{Y}_1])\\ &\qquad (a_2(\mathrm{X}_2 - E[\mathrm{X}_2]) + b_2(\mathrm{Y}_2 - E[\mathrm{Y}_2])].\end{aligned} \tag{51}$$

Rearranging above equation and using (45) led to the final result for the covariance as follows:

$$\begin{aligned}\mathrm{cov}(\mathrm{Z}_1, \mathrm{Z}_2) &= a_1a_2\mathrm{cov}(\mathrm{X}_1, \mathrm{X}_2) + b_1\,b_2\mathrm{cov}(\mathrm{Y}_1, \mathrm{Y}_2)\\ &\quad + a_1\,b_2\mathrm{cov}(\mathrm{X}_1, \mathrm{Y}_2) + a_2\,b_1\mathrm{cov}(\mathrm{X}_2, \mathrm{Y}_1).\end{aligned} \tag{52}$$

REFERENCES

[1] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.
[2] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, and D. Johnston, "Junior: The stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, Sep. 2008.
[3] S. Kolski, D. Ferguson, M. Bellino, and R. Siegwart, "Autonomous driving in structured and unstructured environments," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2006, pp. 558–563.
[4] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, Apr. 2010.
[5] J. Petereit, T. Emter, C. W. Frey, T. Kopfstedt, and A. Beutel, "Application of Hybrid A* to an autonomous mobile robot for path planning in unstructured outdoor environments," in *Proc. 7th German Conf. Robot. (ROBOTIK)*, 2012, pp. 1–6.
[6] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, 2010.
[7] S. Koenig and M. Likhachev, "D* lite," in *Proc. AAAI/IAAI*, vol. 15, 2002, pp. 476–483.
[8] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," in *Proc. ICAPS*, vol. 5, 2005, pp. 262–271.
[9] D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," in *Robotics Research* Berlin, Germany: Springer, 2007, pp. 239–253.
[10] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," 2017, *arXiv:1702.01182*. [Online]. Available: http://arxiv.org/abs/1702.01182
[11] G. Kahn, P. Abbeel, and S. Levine, "BADGR: An autonomous self-supervised learning-based navigation system," 2020, *arXiv:2002.05700*. [Online]. Available: http://arxiv.org/abs/2002.05700
[12] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Ames, IA, USA, Tech. Rep., 1998.
[13] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[14] I. Ardiyanto and J. Miura, "Real-time navigation using randomized kinodynamic planning with arrival time field," *Robot. Auto. Syst.*, vol. 60, no. 12, pp. 1579–1591, Dec. 2012.
[15] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 723–730.
[16] N. A. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 1617–1624.
[17] L. Blackmore, H. Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles," in *Proc. Amer. Control Conf.*, 2006, p. 7.
[18] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Trans. Robot.*, vol. 27, no. 6, pp. 1080–1094, Dec. 2011.
[19] B. Luders, M. Kothari, and J. How, "Chance constrained RRT for probabilistic robustness to environmental uncertainty," in *Proc. AIAA Guid., Navigat., Control Conf.*, Aug. 2010, p. 8160.

[20] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Auto. Robots*, vol. 35, no. 1, pp. 51–76, Jul. 2013.
[21] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
[22] B. D. Luders, S. Karaman, and J. P. How, "Robust sampling-based motion planning with asymptotic optimality guarantees," in *Proc. AIAA Guid., Navigat., Control (GNC) Conf.*, Aug. 2013, p. 5097.
[23] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A probabilistic particle-control approximation of chance-constrained stochastic predictive control," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 502–517, Jun. 2010.
[24] B. Luders and J. P. How, "Probabilistic feasibility for nonlinear systems with non-Gaussian uncertainty using RRT," in *Proc. AIAA Infotech@Aerosp. Conf.*, St. Louis, MO, USA, Mar. 2011, p. 1589.
[25] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 895–913, Jun. 2011.
[26] C. Chen, M. Rickert, and A. Knoll, "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2015, pp. 1148–1153.
[27] C. Chen, M. Rickert, and A. Knoll, "Motion planning under perception and control uncertainties with space exploration guided heuristic search," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 712–718.
[28] Y. Zhang, H. Chen, S. L. Waslander, J. Gong, G. Xiong, T. Yang, and K. Liu, "Hybrid trajectory planning for autonomous driving in highly constrained environments," *IEEE Access*, vol. 6, pp. 32800–32819, 2018.
[29] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.
[30] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, Mar. 1960.
[31] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, Oct. 1990.
[32] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767
[33] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[34] C. Scholler, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1696–1703, Apr. 2020.
[35] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Probability*. Belmont, MA, USA: Athena Scientific, 2002.
[36] B. D. Luders, S. Karaman, E. Frazzoli, and J. P. How, "Bounds on tracking error using closed-loop rapidly-exploring random trees," in *Proc. Amer. Control Conf.*, 2010, pp. 5406–5412.
[37] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Kobe, Japan, vol. 3, 2009, p. 5.
[38] A. Lambert, D. Gruyer, and G. S. Pierre, "A fast Monte Carlo algorithm for collision probability estimation," in *Proc. 10th Int. Conf. Control, Autom., Robot. Vis.*, Dec. 2008, pp. 406–411.
[39] J. W. J. Williams, "Algorithm 232: Heapsort," *Commun. ACM*, vol. 7, no. 6, pp. 347–348, Jun. 1964.
[40] D. R. Musser, "Introspective sorting and selection algorithms," *Software: Pract. Exper.*, vol. 27, no. 8, pp. 983–993, Aug. 1997.
[41] C. Ren and S. He, "Finite-time stabilization for positive Markovian jumping neural networks," *Appl. Math. Comput.*, vol. 365, Jan. 2020, Art. no. 124631.
[42] C. Ren, S. He, X. Luan, F. Liu, and H. R. Karimi, "Finite-time L_2-gain asynchronous control for continuous-time positive hidden Markov jump systems via T–S fuzzy model approach," *IEEE Trans. Cybern.*, vol. 51, no. 1, pp. 77–87, Jan. 2021.

TOBIAS RAINER SCHÄFLE received the B.Sc. degree from the University of Stuttgart, Stuttgart, Germany, in 2015, the M.Eng. degree from the Toyohashi University of Technology, Toyohashi, Japan, and the M.Sc. degree from the University of Stuttgart, in 2016. He is currently pursuing the Ph.D. degree with the Toyohashi University of Technology. His research interests include path planning under uncertainties, mobile robot control, and robotics.

NAOKI UCHIYAMA received the (associate) B.E. degree from the Numazu National College of Technology, Shizuoka, Japan, in 1988, and the B.E. and M.E. degrees from Shizuoka University, Shizuoka, in 1990 and 1992, respectively, and the Ph.D. degree in mechanical engineering from Tokyo Metropolitan University, Tokyo, Japan, in 1995. Since 1995, he has been with the Department of Mechanical Engineering, Toyohashi University of Technology, Japan, where he is currently a Professor. He was a Visiting Scholar with the University of California at Davis, Davis, from 2001 to 2002. He is a member of IEEE IES, CSS, and RAS.