Heuristic Search in Belief Space for Motion Planning under Uncertainties

David Lenz[1] and Markus Rickert[1] and Alois Knoll[2]

***Abstract*— In order to fully exploit the capabilities of a robotic systems, it is necessary to consider the limitations and errors of actuators and sensors already during the motion planning phase. In this paper, a framework for path planning is introduced, that uses heuristic search to build up a search graph in belief space, an extension to the deterministic state space considering the uncertainty associated with this space. As sources of uncertainty actuator errors and map uncertainties are considered. We apply this framework to various scenarios for a non-holonomic vehicle and compare the resulting paths to heuristic state space planners and LQG-MP[1] with the help of simulations. As a result, paths generated with this framework could either not be found with worst-case assumptions or have a higher probability of being successfully executed compared to planners with more relaxed constraints.**

I. Introduction

Since the DARPA Grand Challenge in 2005, automated driving has gained more and more focus in academia and with car manufacturers all over the world. All of the research vehicles have in common, that they use very sophisticated sensors like for example the Velodyne Lidar sensor in order to create a very good environment representation around the vehicle. As autonomous driving functions like automated parking, traffic-jam assistants or highway driving are on the verge of becoming available for customers, this approach is not feasible anymore. To keep the costs low and feasible for a product, a cheap and limited sensor set is installed. The mostly noisy signals of the sensors are then usually combined within a probabilistic data fusion framework to get one consistent environment representation and state estimate of the car. The result is a *best guess* on what the environment might look like and can—depending on which sensors see which obstacle—lead to areas which are not well-known but have a large uncertainty. Furthermore, a car especially with a combustion engine is very hard to model and to control such that it behaves exactly the way it was planned beforehand. Fig. 1 shows an example of an obstacle that is known only with noise and also the possible error distribution due to actuators along the reference.

On the other hand, there are motion planning algorithms that need to plan a feasible and safe motion that the vehicle can actually execute. In current systems this is mostly achieved by making conservative worst-case assumptions and adding these to the safety margins used in planning. This limits the true capabilities of a robotic system as most of the time a better performance than worst case can be achieved. Thus, in this paper, a probabilistic model of the sensing and control capabilities of a car is used to leverage all of its capabilities. The proposed framework is not limited to autonomous cars but can be applied to arbitrary robotic systems that can be (locally) linearized.

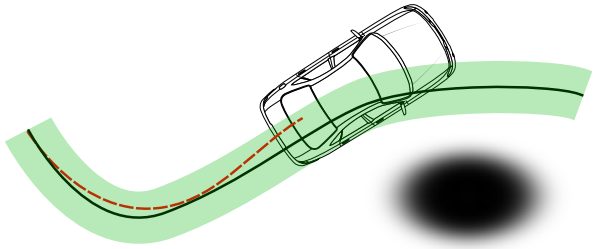

Fig. 1. Reference path with underlying controller errors (green) and one concrete simulation run trajectory (red-dashed). The obstacle has only been observed briefly and is thus not known very well.

II. Related Work

In the past, mostly motion planning under deterministic conditions was considered, but in recent years uncertainties and probabilistic modeling of these have gained a lot of interest. There are four main sources for uncertainties in the planning and in the execution steps, namely:

- Uncertainty in the map due to sensor noise
- Process noise that prevents a perfect execution of motion
- Localization uncertainty due to partial observability
- Unknown future motion of dynamic obstacles

Other sources like modeling errors also exist but can be accounted for in one of these error sources.

On the one hand, there are planners that incorporate *map uncertainty* but treat the robot model as deterministic. For example, Burns and Brock [2] use a utility-guided roadmap planner that minimizes the uncertainty that is encountered along a path and allows utility-guided exploration. Guibas et al. [3] on the other hand consider uncertain corners in a polygonal map and use the success probability in a cost function for a roadmap planner.

For *motion execution errors*, [4] shows a variant of RRT, that uses particles for each tree expansion to represent and calculate the underlying motion uncertainty. Similarly, Alterovitz et al. [5] build a probabilistic roadmap where the edges are associated with a success probability. This is determined by multiple simulations of the execution of the edge with Markov motion uncertainty. Mellinger and Kumar

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

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

[6] use measured error data of a controller in order to obtain a distribution along a path to estimate the collision probability along edges of a search based planner.

In order to additionally consider *partial observability* Platt Jr. et al. [7] formulates the problem of *Belief Space Planning* and apply LQR and nonlinear optimization. Berg et al. [1] on the other hand uses standard RRT to plan multiple times and rank the resulting paths for success probabilities determined with LQG-MP. The work of Bry et al. [8] expands the RRT* algorithm to plan in belief space incorporating Gaussing noise in state measurements and process disturbances. This work is most closely related to the work presented in this paper, but the authors did not consider uncertainty in the map used for planning. Similarly, Vitus and Tomlin [9] show an optimization based framework for linear Guassian system with chance constraints. Berg et al. [10] and Patil et al. [11] use a sequential quadratic programming (SQP) optimization to plan in belief space, but without regarding obstacles at all. Kurniawati et al. [12] use a point-based POMDP with Guided Cluster Sampling to address the problem of high dimensional planning with motion, sensing and environment uncertainty.

Dynamic obstacles are taken into account in [13] in a RRT planner and are modeled according to a pre-learned Gaussian process. Also Du Toit and Burdick [14] focus on dynamic, uncertain behavior of obstacles with a Partially Closed-loop Receding Horizon Control algorithm, that accounts for future information gathering.

In this paper, we focus on the first three parts of uncertainty namely motion, observation and map uncertainties and integrate them into a heuristic search framework in belief space. This allows a guided search of an optimal path within discretization for systems with nonlinear dynamics without the need of a local planning function as many RRT(*) or PRM based planner have. Finally we show with simulation for a Dubins car model that the resulting paths are either safer in execution or better utilize the physical limits of the robot than comparable deterministic planning algorithms. We further show that finding an optimal path is more efficient (in terms of number of edge expansions) due to the heuristic than the LQG-MP [1] approach.

III. Problem formulation

In this section we first define the stochastic motion model, a controller model and the method for uncertainty propagation. Then, the environment model used in this paper is shown. Last, the method for probabilistic collision checking is introduced.

A. Stochastic Motion Model

The stochastic dynamic time-discrete model of a robotic system can be expressed as

$$\begin{aligned} \boldsymbol{x}_{n+1} &= \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{\eta}_n), \quad & \boldsymbol{\eta}_n &\sim N(0, \boldsymbol{\mathcal{M}}_n) \\ \boldsymbol{y}_n &= \boldsymbol{g}(\boldsymbol{x}_n, \boldsymbol{\nu}_n), & \boldsymbol{\nu}_n &\sim N(0, \boldsymbol{\mathcal{N}}_n) \end{aligned} \tag{1}$$

where $\boldsymbol{x}_n \in \boldsymbol{\mathcal{X}}$ denotes the state, $\boldsymbol{u}_n \in \boldsymbol{\mathcal{U}}$ the control input and $\boldsymbol{\eta}_n$ are the error characteristics of the motion model respectively. Further, $\boldsymbol{y}_n$ denotes the state measurement and $\boldsymbol{\nu}_n$ is the error characteristic of the observation model.

The (possibly nonlinear) model described in (1) can be linearized around a nominal trajectory $(\boldsymbol{x}_n^*, \boldsymbol{u}_n^*, \boldsymbol{\eta}_n^* = 0)$ to:

$$\begin{aligned} \tilde{\boldsymbol{x}}_{n+1} &= \boldsymbol{A}_n \tilde{\boldsymbol{x}}_n + \boldsymbol{B}_n \tilde{\boldsymbol{u}}_n + \boldsymbol{\eta}_n \\ \tilde{\boldsymbol{y}}_n &= \boldsymbol{C}_n \tilde{\boldsymbol{x}}_n + \boldsymbol{\nu}_n \end{aligned} \tag{2}$$

with $\boldsymbol{A}_n = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_n}|_{(\boldsymbol{x}_n^*, \boldsymbol{u}_n^*)}$, $\boldsymbol{B}_n = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_n}|_{(\boldsymbol{x}_n^*, \boldsymbol{u}_n^*)}$, $\boldsymbol{C}_n = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_n}|_{\boldsymbol{x}_n^*}$ and the state difference $\tilde{\boldsymbol{x}}_n = \boldsymbol{x}_n - \boldsymbol{x}_n^*$. Note that in the rest of the paper, the differences $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{y}}$ and the true values $\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}$ are used interchangeably, as they can easily be converted with the known reference.

As the state cannot be measured exactly, we will introduce the state estimate with mean $\hat{\boldsymbol{x}}$ and covariance of $\boldsymbol{\Sigma}$. It is calculated with a Kalman-filter based on the measurement that are expected to be made. The mean $\hat{\boldsymbol{x}}$ itself will usually not lie on the reference and is further Gaussian distributed with covariance of $\boldsymbol{\Lambda}$.

Definition 3.1: A **belief state** $\boldsymbol{b}_n$ describes the probability distribution of possible states. The probability distribution function $P_{\boldsymbol{b}}(\boldsymbol{b}_n = \boldsymbol{x}_n)$ returns the probability that a belief state $\boldsymbol{b}_n$ takes the value of the state $\boldsymbol{x}_n$.

In this paper, a belief state $\boldsymbol{b}_n$ is encoded as a multi-variate Gaussian with mean $\boldsymbol{x}_n$, the covariance of the state estimate $\boldsymbol{\Sigma}_n$ and the covariance of the mean of the state estimate $\boldsymbol{\Lambda}_n$.

B. Controller Model

Only integrating (2) with a nominal input $\boldsymbol{u}_n^*$ will lead to over-conservative estimates of the belief state. The reason is, that at each stage, a controller can in fact adjust the input to reduce the error around the nominal trajectory. Thus we choose

$$\boldsymbol{u}_n := -\boldsymbol{K}_n \hat{\boldsymbol{x}}_n \tag{3}$$

with an arbitrary stabilizing linear state-feedback control matrix $\boldsymbol{K}_n$.

C. Uncertainty Propagation

Following [8], we treat the observation that will be made as a random variable and keep track of the distribution of means of the Kalman Filter. This means that at time n we expect to take a measurement with covariance of $\boldsymbol{\mathcal{N}}_n$. Because of that, the state estimate $\hat{\boldsymbol{x}}_n$ will usually not lie on the reference trajectory but as mentioned before will be distributed with covariance $\boldsymbol{\Lambda}$.

First, we predict the covariance around $\hat{\boldsymbol{x}}_n$ with a Kalman filter prediction step as

$$\bar{\boldsymbol{\Sigma}}_{n+1} = \boldsymbol{A}_n \boldsymbol{\Sigma}_n \boldsymbol{A}_n^\intercal + \boldsymbol{\mathcal{M}}_n. \tag{4}$$

and with the update step, we get the final covariance

$$\begin{aligned} \boldsymbol{S}_n &= \boldsymbol{C}_n \bar{\boldsymbol{\Sigma}}_n \boldsymbol{C}_n^\intercal + \boldsymbol{\mathcal{N}}_n \\ \boldsymbol{L}_n &= \bar{\boldsymbol{\Sigma}}_n \boldsymbol{C}_n^\intercal \boldsymbol{S}_n^{-1} \\ \boldsymbol{\Sigma}_n &= \bar{\boldsymbol{\Sigma}}_n - \boldsymbol{L}_n \boldsymbol{C}_n \bar{\boldsymbol{\Sigma}}_n. \end{aligned} \tag{5}$$

The covariance of the state estimate can be calculated with the recursive formula:

$$\boldsymbol{\Lambda}_{n+1} = (\boldsymbol{A}_n - \boldsymbol{B}_n\boldsymbol{K}_n)\boldsymbol{\Lambda}_n(\boldsymbol{A}_n - \boldsymbol{B}_n\boldsymbol{K}_n)^\intercal + \boldsymbol{L}_n\boldsymbol{C}_n\bar{\boldsymbol{\Sigma}}_n \tag{6}$$

The final resulting distribution at time n for the state from the reference and its estimate is

$$\begin{pmatrix}\boldsymbol{x}_n \\ \hat{\boldsymbol{x}}_n\end{pmatrix} \sim N\left(\begin{pmatrix}\boldsymbol{x}_n^* \\ \boldsymbol{x}_n^*\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Lambda}_n + \boldsymbol{\Sigma}_n & \boldsymbol{\Lambda}_n \\ \boldsymbol{\Lambda}_n & \boldsymbol{\Lambda}_n\end{pmatrix}\right). \tag{7}$$

The covariance $\boldsymbol{\Lambda}_n + \boldsymbol{\Sigma}_n$ is the uncertainty that is of interest from a planning perspective and will be further used for collision probability calculations.

For a more detailed derivation of the formulas, the reader is referred to [8].

D. Environment Model

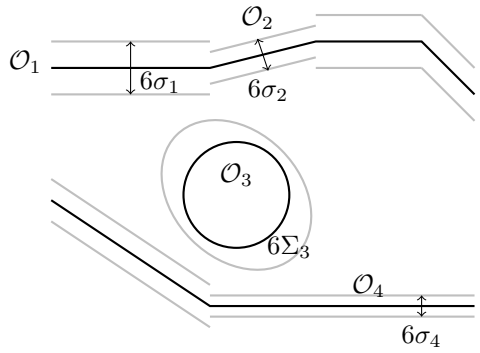

Fig. 2. Example environmental model used in this paper consisting of line segments and circular obstacles. For each object $\mathcal{O}_i$ its associated $6\sigma_i$ covariance interval is shown. The obstacle will be within this interval with a probability of 99.7%

The environment that we regard in this paper consists of a set of N static obstacles $\{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_N\}$ that have an associated uncertainty of position and/or dimension. An example including the confidence intervals of the obstacles can be seen in Fig. 2. Here, circular obstacles and line segments are regarded with Gaussian uncertainty in their position as indicated in the figure.

In contrast to classic planning algorithms we cannot divide the state space $\boldsymbol{\mathcal{X}}$ into valid configurations $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_{free}$ and invalid or colliding configurations $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_c$. But we can only express the probability of the membership to one of the sets as $P_{\text{free}}(\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_{\text{free}})$ and $P_c(\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_c)$. In order to calculate these expressions, the probability that a configuration $\boldsymbol{x}$ collides with an obstacle $\mathcal{O}_i$ will be denoted as

$$P_{c,\mathcal{O}_i}(\boldsymbol{x}) \tag{8}$$

Applying this for each obstacle separately and assuming stochastic independence of all obstacles, we can now calculate the overall collision probability of a configuration as

$$P_c(\boldsymbol{x}) = 1 - \prod_{i=1}^{N}(1 - P_{c,\mathcal{O}_i}(\boldsymbol{x})) \tag{9}$$

E. Probabilistic Collision Checking

As we have seen in the previous section, the function described in (9) gives a collision probability for a given deterministic state, i.e., for a concrete realizations of the random variable. But we are interested in the probability that the state distribution of a belief state collides. In order to calculate this collision probability of a belief state the following joint distribution has to be calculated:

$$P_c(\boldsymbol{b}) = \int_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} P_c(\boldsymbol{x}) \cdot P(\boldsymbol{b} = \boldsymbol{x}) d\boldsymbol{x} \tag{10}$$

This equation is usually not solvable analytically and must be approximated. This can be done for example through conservative approximations as in [8] or through Monte-Carlo sampling [15]. As none of these papers regard the robot distribution and the obstacle distribution simultaneously, we decided to apply sampling. As for stable results many samples are needed this is the most computational demanding step and needs to be addressed for use in an online planning system. This might be possible with a combination of truncated Gaussians [16] and collision probability approximations presented in [17]. Note that it is not possible to first shift the obstacle uncertainty into robot position uncertainty in order to simply analytically compute the probability. The reason is, that the resulting distribution will in general not be Gaussian and even might not have an analytical solution.

IV. Heuristic Search

We propose an algorithm based on the heuristic search A* that we expand to handle belief states and success probabilities. Because of the underlying A* algorithm, the planner will find an optimal solution within the discretization of the input space.

A. Search Algorithm

Algorithm 1: Heuristic search in belief space

```
1  V ← {V_0}, E ← {}, OPENLIST ← {V_0};
2  while !OPENLIST.isEmpty() do
3      V_i ← OPENLIST.popFront();
4      if V_i ∈ X_goal then
5          return success
6      forall the u ∈ U do
7          B_new ← integrateModel(b_i, u);
8          p_s,new ← calculateSuccessProbability(B_new);
9          if p_s,new > p_s,min then
10             g_new ← calculateCost(B_new, V_i);
11             h_new ← heuristic(b_new);
12             (V, E) ← append(V, E, V_new, E_new);
13             OPENLIST.append(V_new);
14     OPENLIST.remove(V_i);
15 return failure;
```

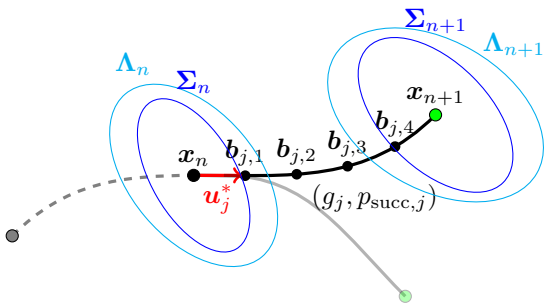

Fig. 3. Part of a search graph during expansion. It shows an expansion from vertex n to vertex $n+1$ along edge j. At the vertices, the belief state $\boldsymbol{b}_n$ is drawn as its mean $\boldsymbol{x}_n$ and the covariances $\boldsymbol{\Sigma}_n$ and $\boldsymbol{\Lambda}_n$

The pseudo-code of the complete algorithm is shown in Algorithm 1. Vertices in the search tree consist of

$$V_n = \{\underbrace{\boldsymbol{x}_n, \boldsymbol{\Sigma}_n, \boldsymbol{\Lambda}_n}_{\boldsymbol{b}_n}, g_n, h_n, p_{\mathrm{s},n}\} \tag{11}$$

where g_n denotes the cost from start to vertex n, h_n the estimated *cost-to-go* (heuristic) and $p_{\mathrm{s},n}$ the probability to successfully reach vertex V_n without collision. Edges on the other hand are defined as

$$E_j = \{\mathcal{B}_j, \boldsymbol{u}_j^*, g_j, p_{\mathrm{s},j}\} \tag{12}$$

with $\mathcal{B}_j$ being a list of belief states, $\boldsymbol{u}_j^*$ the used control input, g_j the cost and $p_{\mathrm{s},j}$ the success probability along edge E_j. A small subset of the search graph with all the used indices can be seen in Fig. 3.

For each new iteration expansion we take the (yet unvisited) graph vertex i with

$$\min_i (g_i + h_i). \tag{13}$$

The current sets of vertices and edges are denoted as $\mathcal{V}$ and $\mathcal{E}$ respectively.

B. Model integration

In order to make a node expansion (edge j) from a given belief state $\boldsymbol{b}_{j,0} = \boldsymbol{b}_n$ to $\boldsymbol{b}_{j,L} = \boldsymbol{b}_{\mathrm{new}}$ with a fixed input $\boldsymbol{u}$ for some expansion time $t = L\Delta t$ we subsequently apply (1) for the state propagation and (4, 5, 6) for the covariance propagation for always Δt. As a result we get a list of belief states.

$$\mathcal{B}_j = \{\boldsymbol{b}_{j,0}, \boldsymbol{b}_{j,1}, \cdots, \boldsymbol{b}_{j,L}\} \tag{14}$$

In order to calculate the success probability $p_{\mathrm{s},j}$ of such an edge, we need to calculate

$$p_{\mathrm{s},j} = P_{\mathrm{s}}(\mathcal{B}_j) = \prod_{l=0}^{L} (1 - P_c(\boldsymbol{b}_{j,l})) \tag{15}$$

again assuming independence of the collision probabilities $P_c(\boldsymbol{b})$. Clearly this time, the assumption is not justified, because if one belief state $\boldsymbol{b}_{j,l}$ is colliding then with a very high probability the subsequent $\boldsymbol{b}_{j,l+1}$ will have a similar probability of collision. As this is an over-approximation of the collision probability this is not a big issue at the moment, but a better approximation can be achieved with truncated Gaussians as in Patil et al. [16].

C. Cost function

The cost function calculateCost() can, in general, be any arbitrary deterministic cost function g_d that is used in conventional motion planners, like for example path length. This would imply that the collision risk of executing a path will only sort out paths that exceed a given threshold of risk. Or put differently, the optimization of the overall cost function will find the path with the lowest cost under the chance constraint that the success probability has to be greater than a threshold.

$$\min_{path} \quad g_d, \quad \text{subject to} \quad p_{\mathrm{s}} \geq p_{\mathrm{s,min}}$$

In order to include the success probability into the cost function and thus have a way to optimize it, we define a new cost

$$g := g_d + \lambda(1 - p_{\mathrm{s}}) \tag{16}$$

with a balancing parameter λ.

D. Heuristic

As with A* planning, the heuristic has to be admissible, that means that it has to always underestimate the true cost-to-go. As seen in section IV-C, all modifications to the deterministic cost function only adds costs, such that $g \geq g_d$. Thus all heuristics that are admissible for a deterministic planner will be also admissible here.

V. Experiments

In order to evaluate the usefulness of our algorithm, we implemented the algorithm of last section in Matlab. We use a circular[1] robot with a radius of $r = 1$ m behaving according to Dubins car model with the state and input vectors

$$\boldsymbol{x} = (x, y, \theta)^T, \boldsymbol{u} = (v, \delta)^T, \boldsymbol{\eta} = (\eta_v, \eta_\delta)^T \tag{17}$$

where x, y are the Cartesian coordinates, θ the orientation of the vehicle, v the commanded velocity and δ denotes the steering curvature. The equation of the dynamics is defined as:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta t \begin{pmatrix} (v + \eta_v)\cos(\theta) \\ (v + \eta_v)\sin(\theta) \\ \delta + \eta_\delta \end{pmatrix} \tag{18}$$

with Δt being the integration time step size. The covariance $\boldsymbol{\mathcal{M}}_n$ of $\boldsymbol{\eta}$ can be time varying or state dependent. Following similar argumentation as in Thrun et al. [18] we assume that the error is only dependent on the currently desired command $\boldsymbol{u}$ as

$$\boldsymbol{\mathcal{M}}_n = \mathrm{diag}(\alpha_v \cdot v_n^2, \alpha_\delta \cdot \delta_n^2 + \alpha_{\delta,v} \cdot v_n^2) \tag{19}$$

with error parameters $\alpha_v = 0.5, \alpha_\delta = 1$ and $\alpha_{\delta,v} = 0.001$. This error definition implies, that higher control inputs can be carried out by the vehicle with less accuracy. As the controller we take a linear feedback control law with an

[1] In order to consider a more realistic vehicle it is possible to approximate its shape with multiple circles

arbitrary stable time-varying feedback matrix, such that the error will always be calculated in track-coordinates and will be invariant to rotation.

$$\boldsymbol{u}_n = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix} \cdot \mathrm{diag}(\cos\theta_n^*, \sin\theta_n^*, 1) \cdot (\boldsymbol{x}_n - \boldsymbol{x}_n^*) \tag{20}$$

As the observation model, we use $\boldsymbol{y}_n = \boldsymbol{x}_n + \boldsymbol{\nu}_n$ with a noise matrix of $\boldsymbol{\mathcal{M}}_n = \mathrm{diag}(0.05, 0.05, 0.02)$.

Finally, for the heuristic search we take the deterministic costs g_d to be the path length and thus, as an admissible heuristic we take the minimal length solution for Dubins car in free space based on [19][2]. The input space used for planning is $\boldsymbol{\mathcal{U}} \in \{(1,0)^T, (1,0.3)^T, (1,-0.3)^T\}$, the chance constraint bound is set to be $p_{\mathrm{s,min}} = 0.8$.

A. Cost-Risk Tradeoff

We want to analyze the influence of the parameter λ of (16) on the resulting paths from the planner and how we can tradeoff cost and risk. Therefore we created an environment with different *uncertain* passages that lead to the goal from the start on the left. In Fig. 4 we can see planned paths for

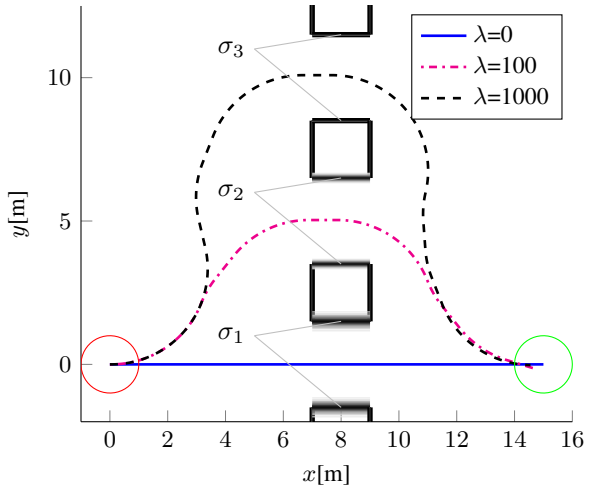

Fig. 4. Analysis of the cost-risk tradeoff parameter λ. The environment consists of three passages with different uncertainties in the width. The covariance for each passage is indicated as $\sigma_1 = 0.2, \sigma_2 = 0.1$ and $\sigma_3 = 0$.

different values for λ within this environment. As expected, depending on the value of λ, the path takes either the more dangerous shorter path or the longer safer through respective passage. Furthermore, we note that for both paths where $\lambda > 0$ the paths are not the minimum length paths. This has to do with the observation (and design of the error covariance) that the controller can achieve a lower covariance on straight paths than on curves. Thus, the planner takes a wider curve and drives straight in order to minimize the covariance within the passage. As a practical interpretation for λ, we are willing to make a detour of 0.1λ meters for a 10% increase in success probability. In the following section, let $\lambda = 100$ as a good tradeoff between cost and risk.

[2]Implementation from: https://github.com/AndrewWalker/Dubins-Curves

B. Comparison of Algorithms

For this experiment, we consider different planning systems:

- *BS-A**: The Belief-Space Heuristic Search presented in this paper
- *ML-A**: An A*-Planner where all obstacles and the robot are assumed to be at the maximum likelihood position. This is the mean for Gaussian distributions and thus we discard the information we have about the covariance.
- *WCO-A**: All obstacles will be set at the worst case and only the robot motion is probabilistic.
- *WC-A**: An A* search where the safety distance is conservative such that the 3σ interval of the worst case deviation from the path will not lead to a collision. The worst case deviation from the path leads to a safety distance of for scenarios I-III $d = 0.1$. As scenario IV can have unbounded uncertainty because the system is not observable.
- *LQG-MP*: From [1] but for collision probability determination we take also the map uncertainty into account and solve the scenario 100 times with a RRT and select the path with the least costs g.

We test all planners on the four environments depicted in Fig. 5. We compare two different metrics: The path length g_d of the found path (if any) and the success probability p_s of the given path. In order to calculate p_s for all paths the same way, we sample the environment, starting conditions and the errors along the path. From there on we can deterministically calculate if the instance of the simulation will collide or not. We repeat the sampling for each path for $M = 1000$ times and count the fraction of runs containing a collision. The results for all scenarios can be found in Table I.

Method	I		II		III		IV	
	g_d	p_s[%]	g_d	p_s[%]	g_d	p_s[%]	g_d	p_s[%]
BS-A*	15	96.4	27	98.5	15	94.0	25.5	96.3
ML-A*	14	72.4	24	21.9	15	60.7	25	36.1
WC-A*	-	-	27	100	-	-	-	-
WCO-A*	15	96.5	27	100	-	-	-	-
LQG-MP	15	99.9	27	98.2	15	93.1	25.5	95.8

TABLE I

SUMMARY OF PATH LENGTH AND ESTIMATED SUCCESS PROBABILITY

1) Scenario I: In this scenario, the task of the robot is to park in a constrained space. The goal position can not be fully seen from the starting pose and thus the uncertainties as shown in Fig. 5(a) arise. As the goal is close to a *blurry* obstacle, the worst case assumption of obstacles and robot position will not find any path in this scenario. For BS-A*, WCO-A* and LQG-MP, all paths look similar and have same length (compare Table I) and a high chance of succeeding. ML-A* on the other hand finds a shorter but unsafer path. With planning in belief space two properties can be seen: Firstly, the path is smoother and especially the final part that approaches the unknown obstacle is straight. Secondly, the path starts off by going straight in order to

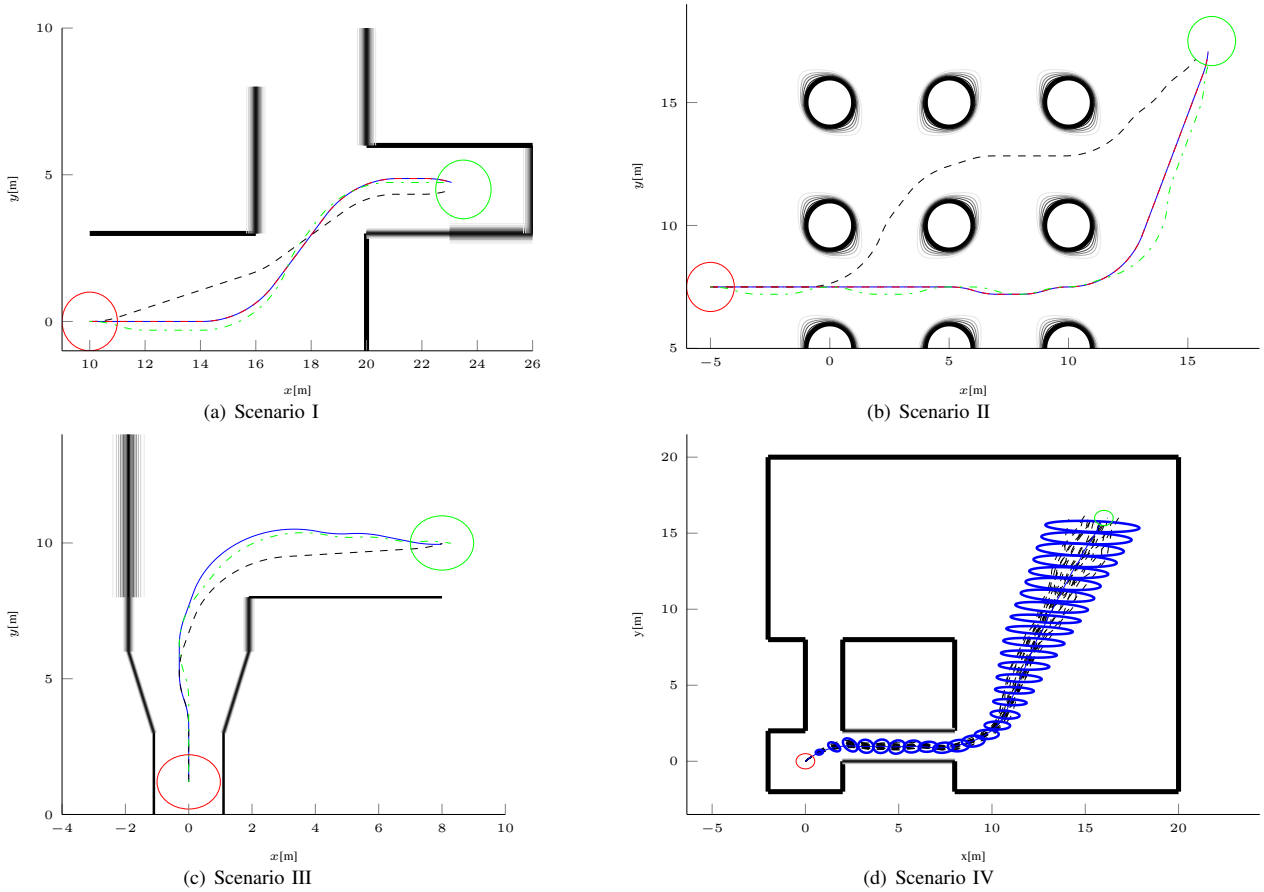

Fig. 5. Sample environments to compare the resulting paths for Belief-Space A*(—), Worst-Case-Obstacles A*(- -), Worst-Case A*(···), Maximum-Likelihood A*(– ·) and LQG-MP (· –). In each figure, the starting position is denoted as a red circle and the goal position as a green circle. (a) Driving into a small passage which cannot be observed from the starting position. WC-A* was not able to plan a path.(b) Navigation through a set of circular obstacles. All planners except ML-A* find similar paths. The path for WC-A* lies under the other two paths. (c) Robot is starting very close to obstacles, only 10 cm in each direction is free space. As already the starting position will collide for the WC-A* no path is shown. (d) Robot is starting in room and can choose between to pathways. Blue ellipses show the covariance $\mathbf{\Sigma} + \mathbf{\Lambda}$ along the path found with BS-A*, black dashed show samples from simulations. As only y coordinate can be measured, uncertainty in x increases and only right path is safe.

better avoid the obstacle in the middle. Both points are a result of the maximization of the success probability, which is also reflected in p_s determined by simulations.

2) Scenario II: Here, a set of circular obstacles is considered. All planners except ML-A* find similar (or even identical) paths, only LQG-MP did not find the optimum after the 100 planning iterations. This is also reflected in the success probability. The probability for success of the path created with ML-A* is with 36.6% not tolerable for a real world system or will lead to constant replanning.

3) Scenario III: The robot starts close to two walls with only 10 cm distance. As everything near the robot is known at this instant perfectly, we are able to plan out of this narrow space. With a worst case position error assumption, this is not possible. Again, the path from BS-A* is much more likely to be successfully executed without replanning. It is noteworthy that taking the map at its worst case position and considering the stochastic motion of the robot with WCO-A* it is not possible to successfully plan this scenario. The reason is, that the first encountered obstacle at its 3σ position prevents finding a path that satisfies the chance constraint $p_{\mathrm{s,min}}$. Thus even though in many cases WCO-A* finds the same path as BS-A* where no or nearly no closeness to obstacles is necessary like in scenario I and II, it is too conservative if some probability of failure has to be taken into account in order to plan.

4) Scenario IV: This scenario is taken from [1] and uses a different observation model than the previous scenarios. Here, we assume only sensing of the y coordinate of the robot, thus $\boldsymbol{y}_n = (0, 1, 0) \cdot \boldsymbol{x}_n + \boldsymbol{\eta}_n$. With this sensing limitation, the only safe path from the starting position is taking the right corridor as the uncertainty in x will grow with time. Similar to the last scenario WC-A* and WCO-A* are not able to find a path. Actually, WC-A* is not applicable as the uncertainty is not bounded because of the observation model and thus there is no such thing as a worst case uncertainty.

5) Complexity: In order to compare the complexity of the presented algorithms, we measure the size of the search graph at the end of planning. As edge expansions are the most expensive operations, we chose the size of edge set $||\mathcal{E}||$ as a measure. Table II shows the result for all presented

Method	I $\|\|\mathcal{E}\|\|$	II $\|\|\mathcal{E}\|\|$	III$\|\|\mathcal{E}\|\|$	IV$\|\|\mathcal{E}\|\|$
BS-A*	362	200	74	257
ML-A*	55	217	75	452
WC-A*	-	162	-	-
WCO-A*	289	169	-	-
LQG-MP (best)	387	108	33	179
LQG-MP (total)	28763	10008	4497	27188

TABLE II

NUMBER OF EDGES IN THE SEARCH GRAPH AFTER PLANNING FINISHED

scenarios and planner. For the LQG-MP algorithm we split up in the search graph for the best (out of 100) and for all planning runs in total. Only BS-A* and LQG-MP produce similar paths but in order to achieve the same result the RRT with LQG-MP has to run multiple times. Thus we have a magnitude higher number of edge expansions. Here, the heuristic that guides the search allows a more efficient way to build up the search graph. In contrast to ML-A*, BS-A* has a factor of up to 6 more expansions, but finds a much safer path. Of course, the cost of one expansion is higher in this case, as the collision check has to determine the probability instead of a binary decision.

VI. CONCLUSION AND FUTURE WORK

In this paper we have shown how the notion of belief space can be integrated into a heuristic search framework in order to successfully plan in an environment with uncertainty in perception and actuation. With this framework it is theoretical possible to integrate all sources of uncertainty mentioned in the introduction, but right now three of them, namely uncertainty in motion, localization and in perception of static obstacles are considered. With this planning algorithm it is possible to plan paths that are more likely to succeed in execution without over-conservative limitations of the vehicles capabilities through worst-case error estimations. Additionally, we obtain paths that

1) can be tracked well by the controller, in this case straight paths
2) stay away from obstacles that are not well known
3) are able to navigate in narrow environments with good knowledge, close to the physical dimensions of the robot
4) consider the localization capabilities

Normally, in motion planning one or multiple of these properties are often achieved by tuning cost functions for example by punishing closeness to obstacles or excessive control inputs. With planning in belief space these desirable behaviors are implicitly created by optimizing success probabilities and exploitation of additional information about errors and uncertainties.

Right now, the implementation is only prototypical and the runtimes are not usable for realtime motion planning yet, mainly due to the inefficient calculation of collision probabilities. In future work, a study of methods for collision probability calculations has to be done towards online application of the framework. Also, dynamic obstacles will be integrated and evaluated in the framework and more thorough comparison to existing planners like [8] will be done. In order to evaluate the practical relevance, experiments with a real autonomous vehicle are planned to determine error characteristics of an implemented tracking controller and observation model. A further interesting field of future work is how to design the heuristic function to include the current (and expected) uncertainty to the goal, for example with

REFERENCES

[1] J. Van Den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics*, vol. 30, no. 7, pp. 895—-913, 2011.

[2] B. Burns and O. Brock, "Sampling-based motion planning with sensing uncertainty," in *IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3313–3318.

[3] L. Guibas, D. Hsu, H. Kurniawati, and E. Rehman, "Bounded uncertainty roadmaps for path planning," *Algorithmic Foundation of Robotics*, vol. VIII, pp. 199–215, 2009.

[4] N. A. Melchior and R. Simmons, "Particle RRT for Path Planning with Uncertainty," in *IEEE International Conference on Robotics and Automation*. IEEE, Apr. 2007, pp. 1617–1624.

[5] R. Alterovitz, T. Siméon, and K. Goldberg, "The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty." *Robotics: Science and Systems*, pp. 246–253, 2007.

[6] D. Mellinger and V. Kumar, "Control and planning for vehicles with uncertainty in dynamics," in *IEEE International Conference on Robotics and Automation*, 2010.

[7] R. Platt Jr., R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Proceedings of the Robotics: Science and Systems Conference, 6th*, 2010.

[8] A. Bry and N. Roy, "Rapidly-exploring Random Belief Trees for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 723–730.

[9] M. Vitus and C. Tomlin, "Closed-loop belief space planning for linear, Gaussian systems," in *IEEE International Conference on Robotics and Automation*, 2011.

[10] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics*, 2012.

[11] S. Patil, Y. Duan, J. Schulman, K. Goldberg, and P. Abbeel, "Gaussian belief space planning with discontinuities in sensing domains," *International Symposium on Robotics*, 2013.

[12] H. Kurniawati, T. Bandyopadhyay, and N. Patrikalakis, "Global motion planning under uncertain motion, sensing, and environment map," *Autonomous Robots*, 2012.

[13] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian processes," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, Sep. 2008, pp. 1056–1062.

[14] N. E. Du Toit and J. W. Burdick, "Robot Motion Planning in Dynamic, Uncertain Environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, Feb. 2012.

[15] A. Lambert, D. Gruyer, and G. Pierre, "A fast Monte Carlo algorithm for collision probability estimation," in *International Conference on Control, Automation, Robotics and Vision*, 2008, pp. 406—-411.

[16] S. Patil, J. van den Berg, and R. Alterovitz, "Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty," in *IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3238–3244.

[17] N. E. Du Toit and J. W. Burdick, "Probabilistic collision checking with chance constraints," *Robotics, IEEE Transactions on*, 2011.

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[19] A. M. Shkel and V. Lumelsky, "Classification of the Dubins set," *Robotics and Autonomous Systems*, vol. 34, no. 4, pp. 179–202, 2001.