An Integrated Localisation, Motion Planning and Obstacle Avoidance Algorithm in Belief Space

Antony Thomas · Fulvio Mastrogiavanni · Marco Baglietto

1 Introduction

Planning and decision making under uncertainty are fundamental requirements for autonomous robots. Uncertainties often arise due to insufficient knowledge about the environment, imperfect sensing and inexact robot motion. In these conditions, the robot poses or other variables of interest can only be dealt with in terms of probabilities. Planning is therefore performed in the belief space, which corresponds to the set of all probability distributions over possible robot states. At a given time instant, we consider the belief or the belief state of the robot which corresponds to a probability distribution of the robot state (or other variables of interest) given the measurements and controls thus far. Consequently, for efficient planning and decision making, it is required to reason about future belief distributions due to candidate actions and the corresponding expected observations. Such a problem falls under the category of Partially Observable Markov Decision Processes (POMDPs).

Robots are becoming ubiquitous in our day-to-day lives and are being increasingly used in close proximity to humans and other objects in service-oriented scenarios such as factories, living spaces, or elderly care facilities. It is therefore of vital importance that robots operate efficiently and safely in real-world conditions. Localization is a key aspect for safe and efficient robot motion as it is a precursor to solving the problems “where to move to” and “how to reach there”. A robot perceives the environment through its sensors and distinct objects known as landmarks aid the robot in localizing. However, most approaches assume that these landmarks are known with high certainty. For example, given the map of the environment, while planning for future actions the standard Markov localization does not take into account the map uncertainty, that is, the landmark location uncertainties are ignored and the locations are assumed to be perfectly known. This means that given the map and the sensor range for any landmark, the application of Bayes filter to the localization problem is called Markov localization.

The application of Bayes filter to the localization problem is called Markov localization.

1 The concepts discussed here are applicable to any sensor used for robot localization. In particular, in this
mark, there exists a set of viewpoints from which an observation may be obtained. Let us consider for example a robot equipped with a laser range finder and observing a landmark. Whenever the robot location is such that the landmark falls within the sensing range, a measurement is obtained. Thus, there exists a set of robot locations or viewpoints from which a measurement of the landmark may be obtained. Therefore, when the landmark locations are assumed to be perfect, this set of viewpoints can be easily determined since it depends on the environment map and the sensing capabilities of the sensor employed. Yet, this might not be true in practice. For example, consider the map of an environment obtained from a Simultaneous Localization and Mapping (SLAM) session. Due to the dynamic nature of the environment, the objects of interests could be occluded when viewed from the set of viewpoints which would have otherwise produced a full observation. Moreover, an erroneous localization, for example due to wrong data association, could lead to wrongly estimated object poses. Thus, in such cases it is more fitting to consider the uncertainty in the landmark locations. This landmark uncertainty directly translates to the fact that the viewpoints whence the object can be observed is uncertain. This is visualized in Fig. 1.

As seen on the left hand side of the figure, when the object location is known perfectly, there exists a region (green) from which the object can be observed. Note that as discussed before this region is determined from the environment map and the sensor capabilities. Some of the viewpoints inside this region are shown in black. On the right hand side of the figure, we consider the uncertainty in landmark location the red shaded region denotes the uncertainty in landmark location. Since the object location is not known precisely (object can be anywhere within the uncertainty region), given a viewpoint, it cannot be said with certainty that the object will be observed. This is so because given a viewpoint, a landmark is observed if it falls within the sensing range. However, since the landmark location is not fixed and is uncertain, the landmark may or may not be within the sensing range. For example, if the landmark location is Gaussian distributed, then the landmark, in practice can be anywhere within the (say) 3-σ uncertainty region. Thus we cannot define a precise region from which the landmark can be observed. Therefore, one can only reason in terms of the probability of observing the object from the considered viewpoint. This results in a probability distribution function for the viewpoints. Consequently, not considering this uncertainty can wrongly localize the robot, leading to inefficient plans causing catastrophes. From now on, we will use the term object uncertainty to refer to the notion of uncertainty in landmark location.

In order to ensure safe robots’ motion, it is also essential to consider collision avoidance strategies. As robots are being increasingly used in service-oriented scenarios with both static and dynamic obstacles, deterministic approaches do not fare well. Moreover, in the case of dynamic obstacles, their future states have to be predicted. Yet this is an added difficulty due to the lack of perfect knowledge of their motions. As a result, providing safety guarantees is difficult.

1.1 Notations and Problem Definition

Throughout this paper vectors will be assumed to be column vectors and will be denoted by lower case letters, that is \( \mathbf{x} \). The transpose of \( \mathbf{x} \) will be denoted by \( \mathbf{x}^T \) and its Euclidean norm by \( \| \mathbf{x} \| = \sqrt{\mathbf{x}^T \mathbf{x}} \). A multivariate Gaussian distribution of \( \mathbf{x} \) with mean \( \mathbf{\mu} \) and covariance \( \Sigma \) will be denoted using the notation \( \mathbf{x} \sim \mathcal{N}(\mathbf{\mu}, \Sigma) \). Matrices will be denoted by capital letters. The trace of a square matrix \( M \) will be denoted by \( \text{tr}(M) \). The identity matrix will be denoted by \( I \) or \( I_n \) when the dimension needs to be stressed. A diagonal matrix with diagonal elements \( \lambda_1, \ldots, \lambda_n \) will be denoted by \( \text{diag}(\lambda_1, \ldots, \lambda_n) \). Sets will be denoted using calligraphic capital letters like \( S \) or \( R \). Unless otherwise mentioned, subscripts on vectors/matrices will be used to denote time indexes and (whenever necessary) superscripts will be used to indicate the robot or the object that it refers to. For example, \( x_i^k \) represents the state of robot \( i \) at time instant \( k \). The notation \( P(\cdot) \) will be used to denote the probability of an event and the probability density function (pdf) will be denoted by \( f(\cdot) \).

![Fig. 1: The red blob denotes an object in the environment. The green region corresponds to the set of viewpoints from which the object can be observed; some of these viewpoints are shown as black dots. On the right hand side, the red shaded region denotes the uncertainty in object location, with the red blob denoting its mean position. The corresponding viewpoint region is visualized as the intersection between different viewpoint regions that correspond to the object being at different locations (left hand side shows one instance of this). The trace of a square matrix \( M \) will be denoted by \( \text{tr}(M) \). The identity matrix will be denoted by \( I \) or \( I_n \) when the dimension needs to be stressed. A diagonal matrix with diagonal elements \( \lambda_1, \ldots, \lambda_n \) will be denoted by \( \text{diag}(\lambda_1, \ldots, \lambda_n) \). Sets will be denoted using calligraphic capital letters like \( S \) or \( R \). Unless otherwise mentioned, subscripts on vectors/matrices will be used to denote time indexes and (whenever necessary) superscripts will be used to indicate the robot or the object that it refers to. For example, \( x_i^k \) represents the state of robot \( i \) at time instant \( k \). The notation \( P(\cdot) \) will be used to denote the probability of an event and the probability density function (pdf) will be denoted by \( f(\cdot) \).]
While deriving the **Belief Space Planning (BSP)** framework to incorporate object uncertainties we will mainly follow the notations and formalisms in [31].

We now formally define the problem that we tackle in this paper. Consider a robot operating in a partially-observable environment. The map of the environment is either known a priori or is built using a standard SLAM algorithm. At any time $k$, we denote the robot pose (or configuration) by $x_k = (x_k, y_k, \theta_k)$, the acquired measurement from objects is denoted by $z_k$ and the applied control action is denoted as $u_k$. Note that by objects we refer to both the landmarks and the obstacles in the environment. We consider a standard motion model with Gaussian noise

$$x_{k+1} = f(x_k, u_k) + w_k \sim N(0, R_k)$$

where $w_k$ is the random unobservable noise, modeled as a zero mean Gaussian. We note that modeling the random unobservable noise variables as Gaussians with zero mean is a common practice in robotics [31]. The objects are detected through the robot’s sensors and, assuming data association is known, the observation model can be written as

$$z_k = h(x_k, O_i^k) + v_k \sim N(0, Q_k)$$

where $O_i^k$ is the detected $i$-th object and $v_k$ is the zero mean Gaussian noise. The function $h(x_k, O_i^k)$ denotes the fact that at time $k$, the measurement $z_k$ is obtained by observing the $i$-th object $O_i^k$ from viewpoint (robot location) $x_k$. In the case of a laser-range finder the function $h$ could be defined as the distance between $x_k$ and the location of the object (or any particular point on the object) $O_i^k$. If we consider the case of a camera, $h$ may be defined as a pinhole projection operator, projecting the object $O_i^k$ onto the image plane. Given the models in [1] and [2], in this paper we focus on two aspects. First, we consider the object uncertainties while localizing the robot. Second, we compute the exact probability of collision under obstacle uncertainty, which is modeled as a Gaussian distribution. Finally, we evaluate our approach in two simulation domains: a 2D mobile robot domain and a 2D manipulator domain. It is to be noted that for the manipulator domain we will be concerned with the collision avoidance of the manipulator’s end-effector.

### 1.2 Related Work

BSP has been researched extensively in the past with applications spanning a variety of areas including autonomous navigation, multi-modal planning, and active SLAM [1][11][16][20][23][29][30][32]. [11] consider object uncertainty since they are planning in an unknown environment and require several measurements to obtain confidence estimates of object locations. Thus they perform active perception, that is, to look for robot actions that enhances information to reduce the object uncertainty. This context is different from ours since we consider a known environment with object uncertainty and focus on active localization incorporating these uncertainties. In [20], the concept of object uncertainty is commented upon (they call it scene uncertainty); however they do not show how it affects the state estimation. Dynamic environments are considered in [1][16] however the landmark/beacon locations are assumed to be known perfectly; [29][30] also consider perfect landmark locations in the context of task and motion planning. Thus most active and passive localization-based approaches focus on robot state uncertainty and assume perfect knowledge about the location of the objects in the environment. However, in practice, the environment is seldom known with high certainty and hence providing formal guarantees for safe navigation is imperative.

Patil et al. [21] estimate the probability of collision under robot state uncertainty by truncating the state distributions. In [3], future state distributions are predicted and the uncertainties are used to compute bounded collision probabilities. Lee et al. [18] use sigma hulls to formulate collision avoidance constraints in terms of the signed distance to the obstacles. Du Toit and Burdick [5], Park et al. [19] compute the collision probability by marginalizing the joint distribution between the robot and obstacle location. The distributions are assumed to be Gaussian and the marginalization is computed with an indicator function that is true under the collision condition. However, since there is no closed-form solution to this formulation, an approximation is assumed. Furthermore, Park et al. compute an upper bound for the collision probability. An approximation is computed using Monte Carlo Integration in [17], albeit computationally intensive. Another impressive work that uses Monte Carlo approach is Monte Carlo Motion Planning (MCMP) [8]. This approach first solves a deterministic motion planning problem with inflated obstacles and then adjusts the inflation to compute a path that is exactly as safe as desired.

Linear chance constraints are used to compute bounded collision-free trajectories with dynamic obstacles in [33]. Axelrod et al. [2] focus exclusively on obstacle uncertainty. They formalize a notion of “shadows”, which are the geometric equivalent of confidence
intervals for uncertain obstacles. The shadows fundamentally give rise to loose bounds but the computational complexity of bounding the collision probability is greatly reduced. Uncertain obstacles are modelled as polytopes with Gaussian-distributed faces in [28]. Planning a collision-free path in the presence of “risk zones” is considered in [27] by penalizing the time spent in these risk zones. Risk contour maps which give the risk information (uncertainties in location, size and geometry of obstacles) in uncertain environments are used in [9] to obtain safe paths with bounded risks. A related approach for randomly moving obstacles is presented in [7]. Formal verification methods have also been used to construct safe plans [4,26].

Most approaches discussed above compute the collision probability along a path by summing or multiplying the probabilities along different waypoints in the path. Boole’s inequality is used to decouple the total probability in terms of individual waypoint probabilities. Such approaches tend to be overly conservative and rather than computing bounded collision probabilities along a path, the bound should be checked for each configuration along a path. Moreover, in most approaches, the collision probability computed along each waypoint is an approximation of the true value. On the one hand, such approximations can overly penalize paths and could gauge all plans to be infeasible. On the other hand some approximations can be lower than the true collision probability values and can lead to synthesizing unsafe plans.

1.3 Contributions

In this paper two main theoretical contributions are presented. First, we incorporate object uncertainties in the BSP planning framework and derive the resulting Bayes filter in terms of the prediction and measurement updates of the Extended Kalman Filter (EKF). The second is the computation of the probability of collision under environment uncertainty. We formulate the collision avoidance constraint as a quadratic form in random variables. This provides an exact expression for the collision probability in terms of a converging infinite series. A notion of safety is also formalized to compute configurations that satisfy the required collision probability bounds.

We make the following assumptions: (1) the uncertainties are modelled using Gaussian distributions; (2) while formulating the collision constraint, we assume that the robot and obstacles have circular geometries. However, this is by no means a limitation and the approach can be extended to objects with different geometries by considering the configuration spaces.

2 Object Uncertainty

In this Section, we focus on a BSP formulation that incorporates object uncertainties, that is, the viewpoints whence the objects can be observed are not precisely known. We define the object space \( O = \{O_i | O \} \) is an object, and \( 1 \leq i \leq |O| \) to be the set of all objects in the environment. The motion and observation models can be written in a probabilistic framework as \( p(x_{k+1} | x_k, u_k) \) and \( p(z_k | x_k, O_k^i) \), respectively. Let us consider that at time \( k \) the robot received a measurement \( z_k \) which was originated by observing object \( O_k^i \). Given an initial distribution \( p(x_0) \), and the motion and observation models \( p(x_{k+1} | x_k, u_k) \) and \( p(z_k | x_k, O_k^i) \), the posterior probability distribution at time \( k \) is the belief \( b[x_k] \) and can be written as \( p(x_k | z_k, O_k^i, z_{0:k-1}, u_{0:k-1}) \), where \( O_k^i \) is the object observed at time \( k \), \( z_{0:k-1} = \{z_0, ..., z_{k-1}\} \) is the sequence of measurements up to \( k-1 \) and \( u_{0:k-1} = \{u_0, ..., u_{k-1}\} \) is the sequence of controls up to \( k-1 \). Using Bayes rule and theorem of total probability, \( b[x_k] \) can be expanded as

\[
p(x_k | z_k, O_k^i, z_{0:k-1}, u_{0:k-1}) = \eta_k p(z_k | x_k, O_k^i) p(O_k^i | x_k) \int_{x_{k-1}} p(x_k | x_{k-1}, u_{k-1}) b[x_{k-1}]\]
\]

(3)

where \( \eta_k = 1/p(z_k | z_{0:k-1}, u_{0:k-1}) \) is the normalization constant and \( b[x_{k-1}] \sim N(\mu_{k-1}, \Sigma_{k-1}) \) is the belief at time \( k-1 \). The term \( p(O_k^i | x_k) \) denotes the probability of observing the object \( O_k^i \) from the pose \( x_k \) and models the object uncertainty. Similarly, given an action \( u_k \), the propagated belief can be written as

\[
b[x_{k+1}] = \int_{x_k} p(x_{k+1} | x_k, u_k) b[x_k] \]
\]

(4)

Given the current belief \( b[x_k] \) and the control \( u_k \), the propagated belief parameters, that is, mean and covariance, can be computed using the standard EKF prediction as

\[
\mu_{k+1} = f(\mu_k, u_k)
\]
\[
\Sigma_{k+1} = F_k \Sigma_k F_k^T + R_k
\]

(5)

where \( F_k \) is the Jacobian of \( f(\cdot) \) with respect to \( x_k \). To compute the posterior belief using EKF update equations, we first need to model the term \( p(O_k^i | x_k) \). In this

\footnote{For example, the approach in [4] computes a value lower than the actual when the robot state covariance is small.}
work we model the object distribution as a Gaussian distribution given by
\[ p(O_k^i|x_k) \sim \mathcal{N}(\mu_{O_k^i}, \Sigma_{O_k^i}) \]

where \( \mu_{O_k^i} \) is the mean viewpoint/pose that corresponds to the maximum probability of observing \( O_k^i \) and \( \Sigma_{O_k^i} \) is the associated covariance.

For convenience we state the probability density function (pdf) of multivariate Gaussian distributions. For \( x \sim \mathcal{N}(\mu, \Sigma) \) the pdf is of the form
\[ p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right) \]
\[ (7) \]
where \( \det(\cdot) \) denotes the determinant. Expanding the right hand side of \([3]\), we have
\[ b[x_{k+1}] = \eta_k^O \exp(-J_{k+1}) \]
where \( \eta_k^O \) contains the non-exponential terms and \( J_{k+1} \) is given by
\[ J_{k+1} = \frac{1}{2}(x_{k+1} - h(\mu_{k+1}) - H_{k+1}(x_{k+1} - \mu_{k+1}))^T \]
\[ \quad \left( x_{k+1} - h(\mu_{k+1}) - H_{k+1}(x_{k+1} - \mu_{k+1}) \right) + \frac{1}{2}(x_{k+1} - \mu_{k+1})^T \Sigma_{O_k^i}^{-1} (x_{k+1} - \mu_{O_k^i}) \]
\[ \quad + \frac{1}{2}(x_{k+1} - \mu_{k+1})^T \Sigma_{k+1}^{-1} (x_{k+1} - \mu_{k+1}) \] \[ (8) \]
where \( H_{k+1} \) is the Jacobian of \( h(\cdot) \) with respect to \( x_{k+1} \). We note that when object uncertainty is not considered, the second term in \([8]\) disappears and the results that we derive below reduce to that of the standard EKF update case. The parameters of this Gaussian can be obtained by taking the first and second derivatives of \( J_{k+1} \) with respect to \( x_{k+1} \).

The term \([10]\) is the inverse of the covariance of \( b[x_{k+1}] \) \([3]\), that is,
\[ \Sigma_{k+1} = \left( H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \Sigma_{O_k^i}^{-1} + \Sigma_{k+1}^{-1} \right)^{-1} \]
\[ (11) \]

Since the mean of \( b[x_{k+1}] \) is the value that minimizes \( J_{k+1} \), it is obtained by equating \([7]\) to zero:
\[ H_{k+1}^T Q_{k+1}^{-1} (z_{k+1} - h(\mu_{k+1}) - H_{k+1}(x_{k+1} - \mu_{k+1})) = \Sigma_{k+1}^{-1} (\mu_{k+1} - \bar{\mu}_{k+1}) - \Sigma_{O_k^i}^{-1} (\mu_{O_k^i} - \bar{\mu}_{k+1}) \]
\[ \Rightarrow \mu_{k+1} = \bar{\mu}_{k+1} + K_{k+1} (z_{k+1} - h(\mu_{k+1})) + \Sigma_{k+1}^{-1} \Sigma_{O_k^i}^{-1} (\mu_{O_k^i} - \bar{\mu}_{k+1}) \] \[ (12) \]
where \( K_{k+1} = \Sigma_{k+1} H_{k+1}^T Q_{k+1}^{-1} \) is the Kalman gain.

As in the case of standard EKF, the gain \( K_{k+1} \) can be transformed to an expression that does not depend on \( \Sigma_{k+1} \), by post-multiplying with an identity matrix \( I = AA^{-1} \), where
\[ A = \left( H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \Sigma_{O_k^i}^{-1} \right)^{-1} \Sigma_{O_k^i} \]
\[ (13) \]
This gives
\[ K_{k+1} = \left( H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \Sigma_{O_k^i}^{-1} \right)^{-1} \Sigma_{O_k^i} H_{k+1}^T + Q_{k+1} \] \[ (14) \]

In order to simplify the above expression for \( K_{k+1} \), we first compute the inverse of the term
\[ \Sigma_{k+1}^{-1} \left( I + \Sigma_{O_k^i} \right)^{-1} \Sigma_{O_k^i} \]
\[ (15) \]
The inverse is computed as
\[ \left( \Sigma_{k+1}^{-1} \left( I + \Sigma_{O_k^i} \right)^{-1} \Sigma_{O_k^i} \right)^{-1} \]
\[ = \Sigma_{O_k^i}^{-1} \left( \Sigma_{k+1} + \Sigma_{O_k^i} \right)^{-1} \Sigma_{k+1}^{-1} \Sigma_{O_k^i} \]
\[ = \Sigma_{O_k^i}^{-1} \Sigma_{k+1} + \Sigma_{O_k^i} \Sigma_{k+1}^{-1} \Sigma_{O_k^i} \] \[ (16) \]
Using \([16]\) and \([14]\), the expression in \([14]\) simplifies to
\[ K_{k+1} = \Sigma_{k+1} \left( H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \Sigma_{O_k^i}^{-1} + \Sigma_{k+1}^{-1} \right)^{-1} \]
\[ = \Sigma_{k+1} \left( H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \Sigma_{O_k^i}^{-1} + \Sigma_{k+1}^{-1} \right)^{-1} \Sigma_{O_k^i} H_{k+1}^T + Q_{k+1} \]
\[ = \Sigma_{k+1} \left( H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \Sigma_{O_k^i}^{-1} + \Sigma_{k+1}^{-1} \right)^{-1} \Sigma_{O_k^i} H_{k+1}^T \]
\[ (17) \]
By treating the sum $\Sigma_{O_{k+1}^{-1}} + \Sigma_{k+1}^{-1}$ in (11) as a single term and applying the matrix inversion lemma on the right hand side of (11) and further simplifying using the expression for the inverse computed in (16), it can be shown that

$$\Sigma_{k+1} = (I - K_{k+1}H_{k+1}) \Sigma_{k+1} \left( \Sigma_{k+1} + \Sigma_{O_{k+1}^{-1}} \right)^{-1} \Sigma_{O_{k+1}}^{-1} \Sigma_{k+1}^{-1}$$

We note that when no object uncertainty is considered the update step of the standard EKF gives $\mu_{k+1} = \mu_{k+1} + K_{k+1} (z_{k+1} - h(\mu_{k+1}))$ and $\Sigma_{k+1} = (I - K_{k+1}H_{k+1}) \Sigma_{k+1}$. The additional term in (12) rightly adjusts the mean $\mu_{k+1}$ accounting for the fact that the object location is uncertain. Similarly, the extra terms in (13) account for the object uncertainty and scale the posterior covariance accordingly.

### 3 Collision Probability

Let $\mathcal{R}$ represent the set of all points occupied by a rigid-body robot at any given time. Thus, $\mathcal{R}$ represents the collection of points that form the rigid-body robot. Similarly, let $\mathcal{S}$ represent the set of all points occupied by a rigid-body obstacle. A collision occurs if $\mathcal{R} \cap \mathcal{S} \neq \{\emptyset\}$ and we denote the probability of collision as $P(\mathcal{R} \cap \mathcal{S} \neq \{\emptyset\})$. We assume circular geometries for $\mathcal{R}$ and $\mathcal{S}$ with radii $r_1$ and $s_1$, receptively and we denote the center of mass of the robot and the obstacle by $\mathbf{x}_k$ and $\mathbf{s}$, receptively. By abuse of notation we will use $\mathbf{x}_k$ and $\mathbf{s}$ equivalently to $\mathcal{R}$ and $\mathcal{S}$. The collision condition will be written in terms of the center of mass as $\mathcal{C}_{x_k,s} : \mathcal{R} \cap \mathcal{S} \neq \{\emptyset\}$. It is noteworthy that both $\mathbf{x}_k$ and $\mathbf{s}$ are not known precisely but can only be estimated probabilistically, as seen in the previous section. At this point we would like to stress the fact that the concepts and the derivations herein are valid for any 2D rigid-body robot. A mobile robot may be represented by a minimum area enclosing circle. In the case of a 2D manipulator robot each link can be approximated by bounding circles that tightly enclose the link. For such robots, the collision with an obstacle has to be checked for each bounding circle. For example, consider a manipulator robot with $l$ bounding circles. Then the collision condition for the $i$-th circle ($1 \leq i \leq l$) is given by $\mathcal{C}_{x_k,s}^i$, where $\mathbf{x}_k^i$ is the center of the $i$-th circle.

Let us now consider an obstacle at any given time instant, distributed according to the Gaussian $\mathbf{s} \sim \mathcal{N}(\bar{s}, \Sigma_s)$, where $\bar{s}$ represents the mean and $\Sigma_s$ the uncertainty in the estimation of the object. Given the belief at time $k$, that is, $b[\mathbf{x}_k]$, the probability of collision is given by

$$P(\mathcal{C}_{x_k,s}) = \int_{\mathbf{s}} \int_{\mathbf{x}_k} I_c(\mathbf{x}_k, \mathbf{s}) p(\mathbf{x}_k, \mathbf{s})$$

where $C_{x_k,s}$ as defined above represents the fact that robot configuration $\mathbf{x}_k$ and its collision with obstacle at location $\mathbf{s}$ is considered, and $I_c$ is an indicator function defined as

$$I_c(\mathbf{x}_k, \mathbf{s}) = \begin{cases} 1 & \text{if } \mathcal{R} \cap \mathcal{S} \neq \{\emptyset\} \\ 0 & \text{otherwise.} \end{cases}$$

Du Toit and Burdick [5], Park et al. [19] approximate the integral in (19) as $V p(x_k, s)$, where $V$ is the 2D footprint (area) occupied by the robot. For this approximation, in [5] it is assumed that the robot radius $\varepsilon$ is negligible and a point obstacle is considered for this derivation.

To do away with this approximation, we formulate the above problem by considering an alternative approach. Since the robot and obstacle are assumed to be spherical objects, the collision constraint can be written as

$$\|\mathbf{x}_k - \mathbf{s}\|^2 \leq (r_1 + s_1)^2$$

where $\mathbf{x}_k$ and $\mathbf{s}$ are the random vectors that denote the robot and obstacle pose respectively. Here, $\mathbf{x}_k$ and $\mathbf{s}$ corresponds to the body-fixed frames in the global frame. As noted before, the two random vectors in (19) are distributed according to $\mathbf{s} \sim \mathcal{N}(\bar{s}, \Sigma_s)$ and $\mathbf{x}_k \sim \mathcal{N}(\mu_k, \Sigma_k)$. Let us denote by $W = \mathbf{x}_k - \mathbf{s}$, the difference between the two random variables. Then we know that $\mathbf{w}$ is also a Gaussian, distributed as $\mathbf{w} \sim \mathcal{N}(\mu_w - \bar{s}, \Sigma_k + \Sigma_s)$. The collision constraint can now be written as

$$V = \|W\|^2 = \mathbf{W}^T \mathbf{W} \leq (r_1 + s_1)^2$$

where $V$ is a random vector distributed according to the squared $L_2$-norm of $W$. Now, given the probability density function (pdf) of $V$, the collision constraint in (21) reduces to solving the integral

$$P(\mathcal{C}_{x_k,s}) = \int_0^{(r_1+s_1)^2} p(v)$$

where $p(v) = p_V(v = v)$ is the pdf of $V$. It is noteworthy that the above expression is the cumulative distribution function (cdf) of $V$, which is defined as $F_V((r_1+s_1)^2) = P(V \leq (r_1+s_1)^2)$. 

3.1 Quadratic Form in Random Variables

A quadratic form in random variables is defined as \( \mathbf{Q} \).

**Definition 1** Let \( \mathbf{x} = (x_1, \ldots, x_n)^T \) denote a random vector with mean \( \mathbf{\mu} = (\mu_1, \ldots, \mu_n)^T \) and covariance matrix \( \Sigma \). Then the quadratic form in the random variables \( x_1, \ldots, x_n \) associated with an \( n \times n \) symmetric matrix \( \mathbf{A} = (a_{ij}) \) is

\[
\mathbf{Q}(\mathbf{x}) = \mathbf{Q}(x_1, \ldots, x_n) = \mathbf{x}^T \mathbf{A} \mathbf{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j \tag{24}
\]

Let us define \( \mathbf{y} = \Sigma^{-\frac{1}{2}} \mathbf{x} \) and define a random vector \( \mathbf{z} = (\mathbf{y} - \Sigma^{-\frac{1}{2}} \mathbf{\mu}) \). The resulting distribution of \( \mathbf{z} \) is thus zero mean with covariance being the identity matrix. Thus the quadratic form becomes

\[
\mathbf{Q}(\mathbf{x}) = \left( \mathbf{z} + \Sigma^{-\frac{1}{2}} \mathbf{\mu} \right)^T \Sigma^\frac{1}{2} \mathbf{A} \Sigma^\frac{1}{2} \left( \mathbf{z} + \Sigma^{-\frac{1}{2}} \mathbf{\mu} \right) \tag{25}
\]

Suppose there exists an orthogonal matrix \( \mathbf{P} \), that is, \( \mathbf{P} \mathbf{P}^T = \mathbf{I} \) which diagonalizes \( \Sigma^\frac{1}{2} \mathbf{A} \Sigma^\frac{1}{2} \), then \( \mathbf{P}^T \Sigma^\frac{1}{2} \mathbf{A} \Sigma^\frac{1}{2} \mathbf{P} = \text{diag}(\lambda_1, \ldots, \lambda_n) \), where \( \lambda_1, \ldots, \lambda_n \) are the eigenvalues of \( \Sigma^\frac{1}{2} \mathbf{A} \Sigma^\frac{1}{2} \). The quadratic form can now be written as

\[
\mathbf{Q}(\mathbf{x}) = \left( \mathbf{z} + \Sigma^{-\frac{1}{2}} \mathbf{\mu} \right)^T \Sigma^\frac{1}{2} \mathbf{A} \Sigma^\frac{1}{2} \left( \mathbf{z} + \Sigma^{-\frac{1}{2}} \mathbf{\mu} \right) \tag{26}
\]

where \( \mathbf{u} = \mathbf{P}^T \mathbf{z} = (u_1, \ldots, u_n)^T \) and \( \mathbf{b} = \mathbf{P}^T \Sigma^{-\frac{1}{2}} \mathbf{\mu} = (b_1, \ldots, b_n)^T \). The expression in \( \mathbf{Q}(\mathbf{x}) \) can be written concisely,

\[
\mathbf{Q}(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} = \sum_{i=1}^{n} \lambda_i (u_i + b_i)^2 \tag{27}
\]

**Theorem 1** The cdf of \( \mathbf{Q}(\mathbf{x}) = \mathbf{y} = \mathbf{x}^T \mathbf{A} \mathbf{x} \) with \( \mathbf{A} > 0, \mathbf{x} \sim \mathcal{N}(\mathbf{\mu}, \Sigma) \) is

\[
F_\mathbf{Q}(y) = P(\mathbf{y} \leq y) = \sum_{k=0}^{\infty} (-1)^k c_k \frac{y^{\frac{2}{2} + k}}{\Gamma\left(\frac{2}{2} + k + 1\right)} \tag{28}
\]

and its pdf is given by

\[
p_\mathbf{Q}(y) = P(\mathbf{y} = y) = \sum_{k=0}^{\infty} (-1)^k c_k \frac{y^{\frac{2}{2} + k - 1}}{\Gamma\left(\frac{2}{2} + k\right)} \tag{29}
\]

where \( c_k \) denotes the gamma function and

\[
c_0 = \exp\left(-\frac{1}{2} \sum_{i=1}^{n} b_i^2 \prod_{i=1}^{n} (2\lambda_i)^{-\frac{1}{2}} \right)
\]

\[
c_k = \frac{1}{k} \sum_{i=0}^{k-1} d_{k-i} c_i
\]

\[
d_k = \frac{1}{2} \sum_{i=1}^{n} \left(1 - kb_i^2\right) (2\lambda_i)^{-k}
\]

The proof of the above theorem is beyond the scope of this paper and we refer the interested readers to \( \cite{24} \). It is easily seen that the left hand side of \( \cite{22} \) is in the quadratic form \( \mathbf{Q}(\mathbf{y}) \) with \( \mathbf{A} = I \), the identity matrix. Thus the collision probability can be computed from \( \cite{28} \) as

\[
P(\mathcal{C}_{\mathbf{x}, \mathbf{z}}) = F_\mathbf{y}\left((r_1 + s_1)^2\right) \tag{30}
\]

3.2 Convergence and Truncation Error

In this section we will prove the convergence the infinite series in \( \cite{28} \) and \( \cite{29} \). Note that the series expansion of the pdf in Theorem 1 is of the form

\[
p_\mathbf{y}(y) = \sum_{k=0}^{\infty} c_k h_k(y) \tag{31}
\]

From \( \cite{14} \) we have the following lemma.

**Lemma 1** Let \( \{h_k\}_0^\infty \) be a sequence of measurable complex valued functions on \( [0, \infty) \) and \( \{c_k\}_0^\infty \) be a sequence of complex numbers such that

\[
\sum_{k=0}^{\infty} |c_k|h_k(y)| \leq \alpha e^{\beta y} \quad \text{for } y \in [0, \infty]
\]

where \( \alpha, \beta \) are real constants. Then \( L(h_k(y)) \) and \( L(p_\mathbf{y}(y)) \) exist for \( Re(s) > \beta \), and

\[
L(p_\mathbf{y}(y)) = \sum_{k=0}^{\infty} c_k L(h_k(s)) \tag{33}
\]

where \( L(\cdot) \) denotes the Laplace transform. Let us now define the term \( M(\theta) \) such that

\[
M(\theta) = \sum_{k=0}^{\infty} c_k \theta^k \tag{34}
\]

where the infinite series is a uniformly convergent series for \( \theta \) in some region with \( M(\theta) > 0 \). Let the Laplace transform of \( h_k(y) \) be the form \( L(h_k(y)) = \xi(s)\eta^k(s) \).
where, for \( Re(s) > \beta \) with \( \beta \) being a real constant, \( \xi(s) \) is a non-vanishing analytic function and \( q(s) \) is an analytic function with an inverse function \( \eta(\xi(s)) = \theta \). For \( h_k(y) \) in \( (29) \), we have, \( \xi(s) = (2s)^{-n/2}, \eta(s) = -(2s)^{-1} \) and \( \xi(\theta) = -(2\theta)^{-1} \). Now let us define,

\[
M(\theta) = \left(L(p_\theta) \circ \zeta / \xi \circ \zeta \right)(\theta) = \sum_{k=0}^{\infty} c_k \theta^k
\]

(35)

where \( \circ \) denotes function composition. Using Cauchy's inequality, we get

\[
|c_k| \leq \frac{m(\rho)}{\rho^k}, \quad m(\rho) = \max_{|\theta| = \rho} |M(\theta)|
\]

(36)

Since \( h_k(y) \) is bounded and, using \( (32) \), the condition \( (32) \) in Lemma \( 1 \) is satisfied and the series \( p_\theta(y) \) converges uniformly in every bounded interval of \( y > 0 \). As a result, integrating \( p_\theta(y) \) term-by-term, the obtained series \( F_\theta(y) \) is uniformly convergent in every bounded interval of \( y > 0 \).

If the series in \( (29) \) is truncated after \( N \) terms, the truncation error is

\[
e(N) = \sum_{k=N+1}^{\infty} \left| c_k h_k(y) \right| = \sum_{k=N+1}^{\infty} \left| c_k \theta^{2k-1} \right| \Gamma\left(\frac{n}{2} + k\right)
\]

(37)

Using \( (36) \), an upper bound for the truncation error can hence be obtained as

\[
e(N) \leq \frac{m(\rho)}{\rho^k} \sum_{k=N+1}^{\infty} \left| \frac{y^{2k-1}}{\Gamma\left(\frac{n}{2} + k\right)} \right|
\]

(38)

where the summation term can be further simplified using the gamma function identity, \( \forall \zeta > 0, \Gamma(\zeta + 1) = \zeta \Gamma(\zeta) \), giving

\[
e(N) \leq m(\rho) \left( \Gamma\left(\frac{n}{2}\right) N! \right)^{-1} \left( \frac{n}{2} \right)^{2-1} \left( \frac{n}{2} \right)^{N+1} \exp\left(\frac{n}{2\rho}\right)
\]

(39)

The truncation error for \( (28) \) is obtained in a similar manner,

\[
E(N) \leq m(\rho) \left( \Gamma\left(\frac{n}{2}\right) (N + 1)! \right)^{-1} \left( \frac{n}{2} \right)^{2} \left( \frac{n}{2} \right)^{N+1} \exp\left(\frac{n}{2\rho}\right)
\]

(40)

The expression for \( m(\rho) \) is obtained from \( (14) \).

\[
m(\rho) = \prod_{j=1}^{n} \lambda_j^{-\frac{n}{2}} \exp\left( -\frac{1}{2} \sum_{j=1}^{n} \frac{b_j^2 \lambda_j}{\lambda_j + \rho} \right) \prod_{j=1}^{n} \left( 1 - \frac{\rho}{\lambda_j} \right)^{-\frac{1}{2}}
\]

(41)

The expression in \( (41) \) is valid only if \( \rho < \lambda_j \) \( (14) \) and hence \( \rho < \min \lambda_j \). Thus we have \( m(\rho) \rightarrow 0 \) with \( \sum_{j=1}^{n} b_j^2 \rightarrow \infty \). The larger the distance from the obstacles and the higher the certainty in the robot and obstacle positions, the greater is the \( b_j \) (see \( (26) \)) value. In such scenarios, convergence is often attained within the first few terms of the series. For a given robot configuration and obstacle parameters, we see that the only varying term in \( (40) \) is \( (y/2\rho)^{N+1}/(N+1)! \) which depends on \( \lambda_j \)'s, that is the eigenvalues of \( \Sigma_k + \Sigma_s \). Clearly, at time instant \( k \), the parameter that influences the convergence is the degree of uncertainty in both the robot and obstacle location, that is, \( \Sigma_k + \Sigma_s \).

The convergence is visualized for different configurations in Fig. 2. The blue and green circles represent a robot and an obstacle, respectively. The red ellipses corresponds to the \( 3\sigma \) uncertainties for different covariances diag(0.04, 0.04), diag(0.08, 0.08), ..., diag(0.74, 0.74). In Fig. 2(a) the robot and the obstacle are touching each other. For each of these covariances, the number of terms for convergence is shown in Fig. 2(b). The worst case corresponds to the covariance of diag(0.04, 0.04), requiring 16 terms for convergence (dashed blue line with spikes in Fig. 2(b)). In Fig. 2(c) the distance between the robot and the obstacle is increased by 0.2m and the covariance diag(0.04, 0.04) needed 12 terms for convergence. The distances are further increased by 0.4m and 0.8m in Fig. 2(g) and their worst case convergences are 9 and 5 respectively as seen in Figs 2(f), (h). The number of terms for worst case convergence that corresponds to covariance diag(0.04, 0.04) and the respective time for collision probability computation are shown in Table 1.

| Configuration | Terms for convergence | Computation time (s) |
|---------------|-----------------------|---------------------|
| A             | 16                    | 0.0142 ± 0.0006     |
| B             | 12                    | 0.0044 ± 0.0004     |
| C             | 9                     | 0.0008 ± 0.0003     |
| D             | 5                     | 0.0004 ± 0.0002     |

3.3 Safe Configuration

In the presence of perception and motion uncertainty, providing safety guarantees for robot motion is imperative. Let us assume that the obstacle position is known with high certainty as a result of perfect sensing. However, since the true state of the robot is not known and
only a distribution of these states can be estimated, collision checking has to be performed for this distribution of states. Moreover, in practice, the observations are noisy and this renders the estimated obstacle location (and shape) uncertain. Hence, this uncertainty should be taken into account while considering collision avoidance.

Given a robot configuration \( x_k \), we define the following notion of \( \epsilon \)-safe configuration.

**Definition 2** A robot configuration \( x_k \) is an \( \epsilon \)-safe configuration with respect to an obstacle location \( s \), if the probability of collision is such that \( P(C_{x_k, s}) \leq 1 - \epsilon \).

For example, a 0.99-safe configuration implies that the probability of this configuration colliding with the obstacle is at most 0.01. We use the sampling based Probabilistic Roadmap (PRM) [13] to compute motion plans. As a result we can only guarantee probabilistic completeness for returning \( \epsilon \)-safe configurations since the PRM motion planner is probabilistically complete [12], that is the probability of failure decays to zero exponentially with the number of samples used in the construction of the roadmap. The failure to find an \( \epsilon \)-safe configuration might be because such a configuration indeed does not exist or simply because there were not enough samples.

3.4 Complexity Analysis

It is known that for \( m \) nodes, the computational complexity of PRM is \( O(m \log m) \) [12]. First let us consider the case of belief space planning over the PRM graph, without computing the collision probabilities. Finding a trajectory to the goal requires performing Bayesian (EKF) update operations. This basically involves performing matrix operations—matrix multiplication and inversion of matrices. For a state of dimension \( n \), the covariance matrix is of dimension \( O(n^2) \). Therefore, each step of the Bayesian update has a complexity of \( O(n^3) \).

If \( T \) denotes the number of time steps in the trajectory, then the overall computational complexity is \( O(n^3 T) \).

Let us now analyze the complexity of collision probability computation. From (40) we see that for each iteration, the truncation error varies with \( (y/2\rho) \). Therefore, to achieve \( E(N) \leq \delta \), for an \( \epsilon \)-safe configuration, \( k = O \left( \log \frac{\delta \rho}{\eta(1 - \epsilon)} \right) \) iterations are required. We note that for each obstacle, the runtime is increased by this factor.

4 Cost Function

At each time instant the robot is required to minimize its control usage and proceed towards the goal \( x_g \), while minimizing its state uncertainty. We quantify the state uncertainty by computing the trace of the marginal covariance of the robot position. As a result, we have the following cost function

\[
c = \|\xi(u_k)\|^2_{M_u} + \|x_k - x^g\|^2_{M_v} + tr \left( \|M_{\Sigma}\|^2_{\Sigma_k} \right) + M_C P(C)
\]
where \( \| x \|_{S} = \sqrt{x^T S x} \) is the Mahalanobis norm, \( M_w, M_p, M_C \) are weight matrices and \( \xi(u_k) \) is a function that quantifies control usage. The choice of weight matrices and the control function vary with application. The term \( tr \left( \| M_x \|_{S}^2 \right) = tr \left( M_x^T S_k M_x \right) \), returns the marginal covariance of the robot location. Therefore, \( M_x = \tau M_C \), where \( \tau \) is a positive scalar and \( M_C \) is a matrix filled with zero or identity entries. \( P(C) \) represents the probability of collision and \( M_C \) penalizes the belief states with higher collision probabilities.

The failure to find an \( \epsilon \)-safe configuration might be because such a configuration indeed does not exist or simply because there was not enough samples in the roadmap. In such scenario the roadmap has to be extended. Different strategies could be implemented to efficiently extend the roadmap but is not the main focus of the current paper. Therefore we follow a straightforward approach to add more samples when an \( \epsilon \)-safe configuration cannot be found. Given a node from which no \( \epsilon \)-safe configuration can be found, a circle of certain radius (half the maximum distance allowed between two edges) is drawn. Samples are then added to the roadmap and the PRM graph is updated until an \( \epsilon \)-safe configuration is found or until time-out.

5 Simulation Results

In this section we first provide a comparison of our approach with \cite{19} and \cite{5}. We then explore the capabilities of our approach in two simulation domains. Performance are evaluated on an Intel® Core i7-6500U CPU@2.50GHz × 4 with 8GB RAM under Ubuntu 16.04 LTS.

5.1 Comparison to Other Approaches

Park et al. \cite{19} approximate the integral in \cite{19} as \( V p(x_k, s) \), where \( V \) is the 2D footprint or area occupied by the robot. For computing \( p(x_k, s) \), they first assume a distribution centered around the obstacle with the covariance being the sum of the robot and obstacle location uncertainties. The collision probability is then computed by finding the \( x_k \) that maximizes \( p(x_k, s) \) and formulate the problem as an optimization problem with a Lagrange multiplier. In \cite{5} the density of the center of the robot is used. For comparing with these approaches, we formulate the problem as given in each of these works.\(^6\) In order to validate the values computed using our approach, we perform numerical integration of the expression in \cite{19}, which gives the exact collision probability value.

Three different cases are considered as shown in Fig. 5. The solid green circle denotes an obstacle of radius 0.5m and its corresponding uncertainty contours are shown as green circles. The solid blue circle denotes a robot of radius 0.3m with the blue circles showing the Gaussian contours. We define a collision probability threshold of 0.1, that is, a 0.9-safe configuration. The collision probability values and the computation times are provided in Table 2. In Fig. 3(a), the robot position known with high certainty and our approach computes collision probability as 4.61% and hence the given configuration is a 0.9-safe configuration. The numerical integral provides the actual value and as seen in Table 2 it is computed to be 4.62%, thus proving the exactness of our method. However, the collision probability computed as given in \cite{19} is 33.26% (almost seven times our value), predicting the configuration to be unsafe. The approach in \cite{5} gave the value of 5.84%, a much tighter upper bound. In Fig. 3(b), there is robot uncertainty along the horizontal axis and the collision probability computed using our approach is 8.22%. The actual value is computed to be 8.25%. As compared to the previous case, the probability has almost doubled. This is quite intuitive as seen from the robot uncertainty spread and hence there is greater chance for intersection between the robot and the obstacle. The value computed using the approach in \cite{19} is 36.31% (4.5 times our value). The approach in \cite{5} also gave a higher value of 14.20%. Unlike the approaches in \cite{5,19} our approach rightly predicts the configuration to be a 0.9-safe configuration. The higher values obtained using our approach, we perform numerical integration of the expression in \cite{19}, which gives the exact collision probability value.

\(^6\) For the comparison, the approaches in \cite{5,19} have been reproduced to the best our understanding and the reproduced codes (including numerical integration and our approach) can be found here—[link to codes].
Table 2: Comparison of collision probability methods.

| Case | Algorithm | Collision probability | Computation time (s) | Feasible |
|------|-----------|-----------------------|----------------------|----------|
| (a)  | Numerical integral | 4.62% | 0.8896 ± 0.0356 | Yes |
|      | Du Toit and Burdick [5] | 5.84% | 0.0026 ± 0.0003 | Yes |
|      | Park et al. [19] | 33.26% | 0.2367 ± 0.2081 | No |
|      | Our approach | 4.61% | 0.0232 ± 0.0024 | Yes |
| (b)  | Numerical integral | 8.25% | 1.2309 ± 0.0298 | Yes |
|      | Du Toit and Burdick [5] | 14.20% | 0.0021 ± 0.0001 | No |
|      | Park et al. [19] | 36.31% | 0.2108 ± 0.3067 | No |
|      | Our approach | 8.22% | 0.0208 ± 0.0021 | Yes |
| (c)  | Numerical integral | 14.82% | 1.2450 ± 0.0301 | No |
|      | Du Toit and Burdick [5] | 0.46% | 0.0019 ± 0.0004 | Yes |
|      | Park et al. [19] | 0.61% | 0.3145 ± 0.4610 | Yes |
|      | Our approach | 14.83% | 0.0271 ± 0.0087 | No |

where \( x_k \) denotes the robot pose at time \( k \) and \( u_k = (\delta_{rot1}, \delta_{trans}, \delta_{rot2}) \) is the applied control. The model assumes that the robot ideally implements the following commands in order: rotation by an angle of \( \delta_{rot1} \), translation of \( \delta_{trans} \) and a final rotation of \( \delta_{rot2} \) orienting the robot in the required direction. The robot accrues translational and rotational errors while executing \( u_k \) and localizes itself by estimating its position using signal measurements from beacons \( b_1, \ldots, b_7 \), which are located at \((x_{b_1}, y_{b_1}), \ldots, (x_{b_7}, y_{b_7})\). The signal strength decays quadratically with the distance to the beacon, giving the following observation model with sensor noise \( v_k \).

\[
z_k = \begin{bmatrix} 1/((x_k - x_b)^2 + (y_k - y_b)^2 + 1) \\ \vdots \\ 1/((x_k - x_b)^2 + (y_k - y_b)^2 + 1) \end{bmatrix} + v_k \tag{44}
\]

We validate our approach in the above discussed environment by varying different parameters, a summary of which is provided in Table 3. Below we detail each of cases considered in Table 3. We first consider the motion planning approach for a point-like robot. The cost function is of the form in (12) with \( M_u = 0.3 \), \( M_g = diag(0.8, 0.8) \), \( M_C = diag(1, 1) \) and \( M_C = 10 \). The underlying PRM graph with 65 nodes is shown in Fig. 5 with the green dots denoting the sampled nodes. The robot, starting from its initial belief state (mean pose denoted by \( S \) in the figure) has to reach the node \( x_g \) (G in the figure), while reducing its uncertainty. The blue triangles denote the beacons that aid in localization. The solid black circles with radius 0.5m, represent obstacles in the environment and the red ellipses denote the 3\( \sigma \) covariances (only the \((x,y)\) portion is shown). Unless otherwise mentioned, in all the experiments, 0.99—safe configurations are solicited and the total planning time is the average time for 25 different runs.

We first consider a case with a point robot and no uncertainty in obstacle location. The planned trajec-

An Integrated Localisation, Motion Planning and Obstacle Avoidance Algorithm in Belief Space

value obtained using numerical integral is 14%

by 97% as compared to cases in Fig. 3(a), (b). Actual

also much more certain, with the uncertainty reduced
touching each other (Fig. 3(c)). The obstacle location is

consider the case of a environment where a mo-

We validate our approach in the above discussed en-

environment by varying different parameters, a summary

of which is provided in Table 3. Below we detail each

cases considered in Table 3. We first consider the

motion planning approach for a point-like robot. The

cost function is of the form in (12) with \( M_u = 0.3 \),

\( M_g = diag(0.8, 0.8) \), \( M_C = diag(1, 1) \) and \( M_C = 10 \).

The robot, starting from its initial belief state

(mean pose denoted by \( S \) in the figure) has to

reach the node \( x_g \) (G in the figure), while reducing its

uncertainty. The blue triangles denote the beacons that

aid in localization. The solid black circles with radius

0.5m, represent obstacles in the environment and the

red ellipses denote the 3\( \sigma \) covariances (only the \((x,y)\)

portion is shown). Unless otherwise mentioned, in all

the experiments, 0.99—safe configurations are solicited

and the total planning time is the average time for 25

different runs.

We first consider a case with a point robot and no

uncertainty in obstacle location. The planned trajec-

ing [5][19] are due to the overly conservative nature of the

estimates.

The approach of Park et al. [19] and [5] assumes that

the robot radius is very small. We also compute the

collision probabilities for a robot and an obstacle with

radius 0.05m each, where the robot and the obstacle are

touching each other (Fig. 3(c)). The obstacle location is

also much more certain, with the uncertainty reduced

touching each other (Fig. 3(c)). The obstacle location is

5.2 2D Environment Domain

We consider the case of a environment where a mo-

bile robot moving in an environment of 30m \( \times \) 20m. A

scaled-down top view is seen in Fig. 4(a). The underly-

ing PRM graph, the start (S in the figure) and goal (G

in the figure) locations can also be seen. The gray cir-

cles denote the obstacles in the environment. Fig. 4(b)

shows a Pioneer P3DX robot at the start location. For

the robot motion model, we consider the following non-

linear dynamics [31]

\[
x_{k+1} = x_k + \delta_{trans} \cos(\theta_k + \delta_{rot1}) \\
y_{k+1} = y_k + \delta_{trans} \sin(\theta_k + \delta_{rot1}) \\
\theta_{k+1} = \theta_k + \delta_{rot1} + \delta_{rot2} \tag{43}
\]
Fig. 4: Simulation environment. (a) Scaled-down (×1/4) top view of the environment with the sampled roadmap and start and goal locations of the robot. (b) Pioneer robot at the starting node of the roadmap.

| Approach | Robot radius | Obstacle uncertainty | Beacon (object) uncertainty | Planned trajectory |
|----------|--------------|-----------------------|-----------------------------|--------------------|
| Our      | Point        | No                    | No                          | Fig. 5(a)          |
| Our      | 0.3 m        | Yes                   | No                          | Fig. 5(b)          |
| Our      | 0.3 m        | No                    | No                          | Fig. 5(a)          |
| Our      | 0.3 m        | Yes                   | No                          | Fig. 5(b)          |
| Our      | 0.3 m        | No                    | No                          | Fig. 5(c)          |
| Our      | 0.3 m        | No                    | Yes                         | Fig. 5(d)          |
| Our      | 0.3 m        | No                    | No (true beacon location)   | Fig. 5(f)          |
| Our      | 0.3 m        | No                    | No (mean beacon location)   | Fig. 5(a)          |

Table 3: Different configurations used for the 2D environment domain.

The trajectory in this case is seen in cyan in Fig. 5(a) with total planning time of 0.0051s(±0.0008s). Please note that the total planning time also includes the collision probability computation time. Next, we consider uncertainty in one of the obstacle location, whose covariance ellipse is shown in gray. The planned trajectory is seen in cyan in Fig. 5(b) and the planning was completed under 0.0279s(±0.0043s). Due to the uncertainty in the obstacle location, the robot takes a longer route to avoid collision. A robot of radius 0.3m and certain (negligible uncertainty) obstacles gave the same trajectory as in Fig. 5(a) with a planning time of 0.0055s(±0.0009s). However, when the obstacle location is uncertain the resulting trajectory is as shown in Fig. 5(c). A change in the trajectory is observed, as compared to the case of a point robot in Fig. 5(b). The planning time in this case is 0.0294s(±0.0047s). It is also worth mentioning that in Fig. 5(b) and (c), the roadmap was updated by adding a node since a 0.99-safe configuration could not be found. The added node is seen in brown, with its coordinates being approximately (9, 11). We also run the case with no obstacle uncertainty and a robot of radius 0.3m using the approach of Park et al. [19]. In this case the planned trajectory is as given in Fig. 5(d). Note that using our approach, the same scenario gives a shorter trajectory (Fig. 5(a)). The longer trajectory computed using the approach in [19] is due to the fact that a loose upper bound is computed for the collision probability. As a result a longer trajectory is obtained. Contrary to this, we compute the exact collision probability and hence a shorter trajectory is synthesized. The same scenario is also run with the approach in [3] and produced a trajectory similar to ours. However, since the uncertainties are significantly lower, the approximate collision probability values computed using [3] are much smaller than the actual values.

Next, we consider the case with uncertainty in the location of the beacons. The considered robot radius is 0.3m with the bottom obstacle being uncertain with covariance diag(0.49, 0.49). Taking object uncertainty into account, the planned trajectory with covariance evolution is as shown in Fig. 5(c). Fig. 5(f), shows the trajectory planned with true beacon locations. The beacons are shown in yellow to denote the true location. Considering only the mean position of the beacons and neglecting the position uncertainty, the planned trajectory is as shown in Fig. 5(a). Actual execution of this would lead to collision with the bottom obstacle. However, executing the planned trajectory obtained by considering the uncertainty in beacon locations does not violate the ε-safety criterion and all the configurations are 0.99-safe.
An Integrated Localisation, Motion Planning and Obstacle Avoidance Algorithm in Belief Space

Fig. 5: Trajectory and the covariance evolution for single planning instantiations are shown. Different cases with obstacle uncertainty for a point robot and a robot of radius 0.3 m are shown in (a), (b), (c) and (d). (e) The planned trajectory when there is uncertainty in beacon locations. (f) True beacon locations are shown in yellow.

It is noteworthy that though we have discussed a 2D environment, the approach directly extends to a mobile robot navigating in a 3D environment. In such domains, the mobile robot may be represented by a minimum volume enclosing sphere. Similarly, the obstacles can also be approximated by their corresponding minimum volume enclosing spheres. Hence the collision condition is the same as given in (21) and therefore the approach discussed in this paper remains valid.

5.3 Laser-grasp Domain

We consider two modified versions of the laser-grasp domain as suggested in [22]. In this domain, a planar robot manipulator must locate and proceed towards a round puck. The state space is the position of the manipulator’s end-effector relative to a grasping point defined directly in front of the puck. Though the end-effector position is assumed to be known completely, the state is not directly observed since the puck position is unknown. Its position can be determined using the laser range finder that points out as a horizontal line from the end-effector. The underlying system dynamics is

$$ f(x_t, u_t) = x_t + u_t $$

where $x \in \mathbb{R}^2$ denotes the state space and $u \in \mathbb{R}^2$ is the end-effector velocity. The cost function is of the form in (42) with $M_x = diag(10, 10)$, $M_g = diag(100, 100)$, $M_Z = diag(10000, 10000)$ and $M_C = 10$.

First, we consider a scenario wherein an additional object is placed that aids in localization. In this scenario, the state is the end-effector position which is not known precisely due to actuation errors. The goal is to place the end-effector directly in front of the puck so as to be able to grasp it. Both the object and the puck can be detected by the horizontal laser. However, the object location is not known exactly and the $3\sigma$ uncertainty ellipse is shown in light blue in Fig. 6(a) and (b). The mean position is visualized by the blue

Fig. 6: Trajectory of the end-effector; green dots denote its mean and the red ellipses denote the covariance matrix. The puck is shown in black and the end-effector is shown to its right. (a) Trajectory and covariance evolution when object uncertainty is not considered and (b) when object uncertainty is considered.

Fig. 7: Green dots denote the mean of the state trajectory and the red ellipses denote the covariance matrix. Mean position of the obstacle at each time instant is visualized in blue. (a) State trajectory and covariance evolution during of-line collision avoidance planning. (b) More information is acquired during online planning, reducing the uncertainty of the obstacle and thereby leading to a change in the planned trajectory.
blob and the yellow blob denotes the actual object location. The red ellipses represent the state covariance at different points along the trajectory. Fig. 8(a) shows the case in which object uncertainty is not considered and the object is assumed to be its mean position. The manipulator moves towards the object first, localizing the end-effector position and then proceeds further to place the end-effector at the grasping point. However, as seen in Fig. 8(a), while executing this plan produced offline, not considering the object uncertainty leads to the collision of the end-effector with the true object (in yellow). When the object uncertainty is considered, the execution of the plan do not lead to collision, as it can be seen in Fig. 8(b). This illustrates the fact that not considering object uncertainty can wrongly localize the robot, leading to catastrophes.

Next, we consider a scenario wherein the state space is the position of the manipulator’s end-effector relative to a grasping point defined directly in front of the puck. The state is not directly observed since the puck position is unknown. However, as soon as the manipulator starts to move, a ball starts to roll in between the manipulator and the puck. The ball follows a Gaussian velocity distribution, and therefore at each time instant, the mean position of the ball and the corresponding uncertainty can be estimated. The mean position of the ball at each time instant is shown in blue in Fig. 8(a) and (b). The green dots denote the mean of the state trajectory. As seen in Fig. 8(a), the manipulator initially moves downwards. However, as the ball comes closer, the manipulator retraces its path and move upwards towards its starting position to avoid collision. This is so because the safety constraint for $\epsilon = 0.99$ is violated. As the ball keeps moving upwards, after a while, it is seen that the manipulator takes a downward action just before reaching its starting position since the configuration is a 0.99-safe configuration.

The scenario in Fig. 8(b) is similar to that of Fig. 8(a). However, it is seen that once the manipulator retraces its path backwards towards the starting position, it takes a downward action much earlier. This is because more information is acquired during online planning and the uncertainty bound on the obstacle changes with time.

The 2D manipulator domain studied here directly extends to 3D manipulator scenarios for both static and mobile manipulators. In the case of static manipulators, the end-effector is approximated as a sphere. Each link is approximated as a set of spheres kept side by side. However, in heavily cluttered environments such an approximation can be computationally intensive since each sphere has to be checked for collision with obstacles. An alternative and effective approach is to consider the minimum-volume enclosing ellipsoid for each link [25]. It is known that for every convex polyhedron, there exists a unique ellipsoid of minimal volume that contains the polyhedron and is called the Löwner–John ellipsoid of the polyhedron [6]. Thus each link can be represented by their corresponding Löwner–John ellipsoids. The distance between two ellipsoids is used to modify the collision condition in (21). For mobile manipulators, the collision condition should also checked for the base as discussed in the 2D robot section.

6 Conclusion

In this paper, we have addressed a novel approach to compute the probability of collision under robot and obstacle pose uncertainties. The collision probability is computed as an infinite series whose convergence is proved. An upper bound for the truncation error is also derived. As shown in Fig. 2 convergence analysis is performed for different configurations and it is seen that our approach is of the order of milliseconds and therefore can be used in online planning. We also provide a comparison with the approaches in [5, 19]. In addition, we incorporate landmark uncertainties in belief space planning and derive the resulting Bayes filter in terms of the prediction and measurement updates of the EKF. Finally, experimental evaluation for a mobile robot scenario and a 2D manipulator is performed to illustrate our approach. We have considered static obstacles in this paper and the immediate future work is to realize the approach in simulated and real-world environments with dynamic obstacles.

References

1. Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M Amato. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. The International Journal of Robotics Research, 33(2):268–304, 2014.
2. Brian Axelrod, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Provably safe robot navigation with obstacle uncertainty. The International Journal of Robotics Research, 37(13-14):1760–1774, 2018.
3. Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In IEEE International Conference on Robotics and Automation, pages 723–730, 2011.
4. Xu Chu Ding, Alessandro Pinto, and Amit Surana. Strategic planning under uncertainties via constrained markov decision processes. In IEEE International Conference on Robotics and Automation, pages 4568–4575, 2013.
5. Noel E Du Toit and Joel W Burdick. Probabilistic collision checking with chance constraints. IEEE Transactions on Robotics, 27(4):809–815, 2011.
6. Martin Grötschel, László Lovász, and Alexander Schrijver. Geometric Algorithms and Combinatorial Optimization. Springer-Verlag, New York, 1988.

7. Astghik Hakobyan, Gyeong Chan Kim, and Insoo Yang. Risk-Aware Motion Planning and Control Using CVaR-Constrained Optimization. IEEE Robotics and Automation Letters, 4(4):3924–3931, 2019.

8. Lucas Janson, Edward Schmerling, and Marco Pavone. Monte Carlo motion planning for robot trajectory optimization under uncertainty. In Robotics Research, pages 343–361. Springer, 2018.

9. Ashkan M Jasour and Brian C Williams. Risk contours map for risk bounded motion planning under perception uncertainties. Robotics: Science and Systems, 2019.

10. Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. Artificial Intelligence, 101(1-2):99–134, 1998.

11. Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. The International Journal of Robotics Research, 32(9-10):1194–1227, 2013.

12. Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research, 30(7):846–894, 2011.

13. Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4):566–580, 1996.

14. Samuel Kotz, Norman L Johnson, and DW Boyd. Series representations of distributions of quadratic forms in normal variables. I. Central case. The Annals of Mathematical Statistics, 38(3):823–837, 1967.

15. Samuel Kotz, Norman L Johnson, and DW Boyd. Series representations of distributions of quadratic forms in normal variables ii. non-central case. The Annals of Mathematical Statistics, 38(3):838–848, 1967.

16. Hanna Kurniawati and Vinay Yadav. An Online POMDP Solver for Uncertainty Planning in Dynamic Environment. In Robotics Research: The 16th International Symposium ISRR, pages 611–629, 2016.

17. Alain Lambert, Dominique Guery, and Guillaume Saint Pierre. A fast Monte Carlo algorithm for collision probability estimation. In 10th IEEE International Conference on Control, Automation, Robotics and Vision, pages 406–411, 2008.

18. Alex Lee, Yan Duan, Sachin Patil, John Schulman, Zoe McCarthy, Jur Van Den Berg, Ken Goldberg, and Pieter Abbeel. Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5660–5667, 2013.

19. Chonhyon Park, Jae S Park, and Dinesh Manocha. Fast and bounded probabilistic collision detection for high-DOF trajectory planning in dynamic environments. IEEE Transactions on Automation Science and Engineering, 15(3):980–991, 2018.

20. Shashank Pathak, Antony Thomas, and Vadim Indelman. A unified framework for data association aware robust belief space planning and perception. The International Journal of Robotics Research, 37(2-3):287–315, 2018.

21. Sachin Patil, Jur Van Den Berg, and Ron Alterovitz. Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty. In IEEE International Conference on Robotics and Automation, pages 3238–3244, 2012.

22. Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. In Proceedings of Robotics: Science and Systems. Zaragoza, Spain, 2010.

23. Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. The International Journal of Robotics Research, 28(11-12):1448–1465, 2009.

24. Serge B Provost and AM Mathai. Quadratic forms in random variables: theory and applications. M. Dekker, 1992.

25. Elon Rimon and Stephen P Boyd. Obstacle collision detection using best ellipsoid fit. Journal of Intelligent and Robotic Systems, 18(2):105–126, 1997.

26. Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. Robotics: Science and Systems, 2016.

27. Oren Salzman, Brian Hou, and Siddhartha Srinivasa. Efficient motion planning for problems lacking optimal substructure. In Twenty-Seventh International Conference on Automated Planning and Scheduling, 2017.

28. Luke Shimanuki and Brian Axelrod. Hardness of 3D Motion Planning Under Obstacle Uncertainty. Workshop on Algorithmic Foundations of Robotics, 2018.

29. Antony Thomas, Fulvio Mastrogiovanis, and Marco Baglietto. Task-Motion Planning for Navigation in Belief Space. In The International Symposium on Robotics Research, 2019.

30. Antony Thomas, Fulvio Mastrogiovanis, and Marco Baglietto. Towards Multi-Robot Task-Motion Planning for Navigation in Belief Space. In European Starting AI Researchers’ Symposium. CEUR, 2020.

31. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics. MIT press, 2005.

32. Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. The International Journal of Robotics Research, 31(11):1263–1278, 2012.

33. Hai Zhu and Javier Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. IEEE Robotics and Automation Letters, 4(2):776–783, 2019.