Linearization in Motion Planning under Uncertainty

Marcus Hoerger[1,2], Hanna Kurniawati[1], Tirthankar Bandyopadhyay[2], and Alberto Elfes[2]

[1] The University of Queensland, St Lucia, Brisbane, QLD, 4072, Australia
{m.hoerger, hannakur}@uq.edu.au

[2] CSIRO, Pullenvale, Brisbane, QLD 4069, Australia

Abstract. Motion planning under uncertainty is essential to autonomous robots. Over the past decade, the scalability of such planners have advanced substantially. Despite these advances, the problem remains difficult for systems with non-linear dynamics. Most successful methods for planning perform forward search that relies heavily on a large number of simulation runs. Each simulation run generally requires more costly integration for systems with non-linear dynamics. Therefore, for such problems, the entire planning process remains relatively slow. Not surprisingly, linearization-based methods for planning under uncertainty have been proposed. However, it is not clear how linearization affects the quality of the generated motion strategy, and more importantly where to and where not to use such a simplification. This paper presents our preliminary work towards answering such questions. In particular, we propose a measure, called *Statistical-distance-based Non-linearity Measure (SNM)*, to identify where linearization can and where it should not be performed. The measure is based on the distance between the distributions that represent the original motion-sensing models and their linearized version. We show that when the planning problem is framed as the Partially Observable Markov Decision Process (POMDP), the difference between the value of the optimal strategy generated if we plan using the original model and if we plan using the linearized model, can be upper bounded by a function linear in SNM. We test the applicability of this measure in simulation via two venues. First, we compare SNM with a negentropy-based Measure of Non-Gaussianity (MoNG) —a measure that has recently been shown to be a suitable measure of non-linearity for stochastic systems [1]. We compare their performance in measuring the difference between a general POMDP solver [2] that computes motion strategies using the original model and a solver that uses the linearized model (adapted from [3]) on various scenarios. Our results indicate that SNM is more suitable in taking into account the effect that obstacles have on the effectiveness of linearization. In the second set of tests, we use a local estimate of SNM to develop a simple on-line planner that switches between using the original and the linearized model. Simulation results on a car-like robot with second order dynamics and a 4-DOFs and 6-DOFs manipulator with torque control indicate that our simple planner appropriately decides if and when linearization should be used.

Keywords: Motion planning, Motion planning under uncertainty, POMDP

1 Introduction

An autonomous robot must be able to compute reliable motion strategies, despite various errors in actuation and prediction on its effect on the robot and its environment, and despite various errors in sensing and its interpretation. Computing such robust strategies is computationally hard even for a 3 DOFs point robot [4,5]. Conceptually, this problem can be solved in a systematic and principled manner when framed as the Partially

Observable Markov Decision Process (POMDP) [6]. POMDP represents the aforementioned errors as probability distributions, and estimates the system's states (following a sequence of actions and observations that the robot has performed and perceived) as a probability distributions called *beliefs*. It then computes the best motion strategy with respect to beliefs rather than with respect to single states, because the actual state is uncertain due to errors in the system's dynamics and sensing. Although the concept of POMDP was proposed since early 1960 [7], only in recent years that POMDP starts to become practical for robotics problems (e.g., [8,9]). This advancement is achieved by trading optimality with approximate optimality for speed and memory. But even then, in general, computing close to optimal strategies for systems with complex non-linear dynamics remains relatively slow.

Several general POMDP solvers —one that does not restrict the type of dynamics and sensing model of the system, nor the type of distributions used to represent uncertainty— can now compute good motion strategies on-line with 1-10Hz update rate for a number of robotics problems [2,10,11,12]. However, their speed degrades when the robot has complex non-linear dynamics. To find a good strategy, these methods simulate the effect of many sequences of actions from different beliefs. A simulation run generally invokes many numerical integrations, and more complex dynamics tends to increase the cost of each numerical integration, which in turn significantly increases the total planning cost of these methods. Of course, this cost will increase even more for problems that require more or longer simulation runs, such as in problems with long planning horizon.

Many linearization based methods have been proposed [3,13,14,15,16]. These methods evaluate the effect of many sequences of actions from different beliefs too, but uses a linearized model of the dynamics and sensing for simulation, so as to reduce the cost of each simulation run. Together with linearization, many of these methods assume that each reachable belief must be a Gaussian distribution. This assumption improves the speed of simulation further, because the subsequent belief after an action is performed and an observation is perceived can be computed by propagating only the mean and covariance matrix. In contrast, the aforementioned general solvers use particle representation and therefore must compute the subsequent belief by propagating each particle. As a result, the linearization-based planners require less time to simulate the effect of performing a sequence of actions from a belief, and therefore can *potentially* find a good strategy faster than the general method. However, it is known that linearization in control and estimation performs well only when the system's non-linearity is "weak" [17]. The question is, what constitute "weak" non-linearity in motion planning under uncertainty? Where will it be useful and where will it be damaging to use linearization (and Gaussian) simplifications?

This paper presents our preliminary work to answer the above questions. We propose a measure of non-linearity, called *Statistical-distance-based Non-linearity Measure (SNM)*, to help identify the suitability of linearization in a given problem of motion planning under uncertainty. SNM is based on the total variation distance between the original dynamics and sensing models, and their corresponding linearized models. It is general enough to be applied to any type of motion and sensing errors, and any linearization technique, regardless of the type of approximation to the true beliefs (e.g., with and

without Gaussian simplification). We showed that the difference between the value of the optimal strategy generated if we plan using the original model and if we plan using the linearized model, can be upper bounded by a function linear in SNM. Furthermore, our experimental results (Section 6) indicate that compared to recent state-of-the-art methods of non-linearity measures for stochastic systems, SNM is more sensitive to the effect that obstacles have on the effectiveness of linearization, which is critical for motion planning.

To further test the applicability of SNM in motion planning, we develop a simple on-liner planner that uses a local estimate of SNM to automatically switch between a general planner[2] that uses the original POMDP model and a linearization-based planner (adapted from [3]) that uses the linearized model. Experimental results on a car-like robot with acceleration control, and a 4-DOFs and 6-DOFs manipulators with torque control indicate that this simple planner can appropriately decide if and when linearization should be used and therefore computes better strategies faster than each of the component planner.

2 Related work

Linearization is a common practice in solving non-linear control and estimation problems. It is known that linearization performs well only when the system's non-linearity is "weak" [17]. To identify the effectiveness of linearization in solving non-linear problems, many non-linearity measure have been proposed in the control and information fusion community.

Many non-linearity measures (e.g., [18,19,20]) have been designed for deterministic systems. For instance, [18] proposed a measure derived from the curvature of the non-linear function. The work in [19,20] compute the measure based on the distance between the non-linear function and its nearest linearization. A brief survey of non-linearity measures for deterministic systems is available in [17].

Only recently more work on non-linearity measures for stochastic systems started to flourish. For instance, [17] extends the measures in [19,20] to be based on the average distance between the non-linear function that models the motion and sensing of the system, and the set of all possible linearizations of the function.

Very recently, [1] proposes a different class of measures, which is based on the distance between distribution over states and its Gaussian approximation, called Measure of Non-Gaussianity (MoNG), rather than based on the non-linear function itself. They assume a passive stochastic systems, and computes the negentropy of the non-linear function of the transformed belief — that is, the non-linearity measure of a belief is computed as the negentropy between the subsequent beliefs and their Gaussian approximations. Their results indicate that this non-Gaussianity measure is more suitable to measure the non-linearity of stochastic systems, as it takes into account the effect that non-linear transformations have on the shape of the transformed beliefs. This advancement is encouraging and we will use this measure as a comparator of SNM. However, for this purpose, this measure must be modified because our system is not passive, and in fact, eventually, we would like to have a measure that can be used to decide what strategy to use (e.g., to use a linearized or a general planner). The exact modifications we made can be found in Appendix B.1

Despite the various non-linearity measures that have been proposed, most are not designed to take into account the effect of obstacles to the non-linearity of the robotic system. Except for MoNG, all of the aforementioned non-linearity measures will have difficulties in taking into account the effect of obstacles, even when these effects are embedded in the motion and sensing models. For instance, curvature-based measures requires the non-linear function to be twice continuously differentiable, but the presence of obstacles is very likely to break the differentiability of the motion model. Furthermore, the effect of obstacles is likely to violate the additive Gaussian error, required for instance by [17]. Although MoNG can potentially take into account the effect of obstacles, it is not designed to. The measure is based on the Gaussian approximation to the subsequent belief. In the presence of obstacles this subsequent belief would have support only in the valid region of the state space, and therefore computing the difference between this subsequent belief and its Gaussian approximation is likely to underestimate the effect of obstacles to the effectiveness of linearization. This is exactly the problem we try to alleviate in our proposed non-linearity measure SNM.

Instead of adopting existing approaches in non-linearity measure, SNM adopts the approach commonly used for sensitivity analysis[21,22] of Markov Decision Processes (MDP) —a special class of POMDP where uncertainty is only in the effect of performing actions. It is based on the statistical distance measure between the true transition dynamics and its perturbed versions. Linearized dynamics can be viewed as a special case of perturbed dynamics, and hence this statistical distance measure can be applied as a non-linearity measure too. We do need to extend these analysis, as they are generally defined for discrete state space and are defined with respect to only the dynamics models (MDP assumes the state of the system is fully observable). Nevertheless, such extensions are feasible and the generality of this measure could help decide which linearization method to use.

3 Problem modelling

In this paper, we consider motion planning problems, in which a robot must move from a given initial state to a state in the goal region while avoiding obstacles. The robot operates inside deterministic, bounded, and perfectly known 2D or 3D environments populated by static obstacles.

The robot's dynamics and sensing are uncertain and are defined as follows. Let $S \subset \mathbb{R}^n$ be the bounded n-dimensional state space, $A \subset \mathbb{R}^d$ the bounded d-dimensional control space and $O \subset \mathbb{R}^l$ the bounded l-dimensional observation space of the robot. The robot evolves according to a discrete-time non-linear stochastic system, which we model in the general form $s_{t+1} = f(s_t, a_t, v_t)$ where f is a known non-linear stochastic dynamic system, $s_t \in S$ is the state of the robot at time t, $a_t \in A$ the control input at time t, and $v_t \in \mathbb{R}^d$ is a random control error. At each time step t, the robot receives imperfect information regarding its current state according to a non-linear stochastic function of the form $o_t = h(s_t, w_t)$ where $o_t \in O$ is the observation at time t and $w_t \in \mathbb{R}^d$ is a random observation error.

We now define the motion planning under uncertainty problem for the above system as a Partially Observable Markov Decision Process (POMDP) problem.

Formally, a POMDP is a tuple $\langle S, A, O, T, Z, R, b_0, \gamma \rangle$. The notations S, A and O are the state, action, and observation spaces. The notation T is a conditional probability

function $p(s' \mid s,a)$ (where $s,s' \in S$ and $a \in A$) that represents uncertainty in the effect of actions, while Z is a conditional probability function $p(o|s,a)$ that represents uncertainty on sensing. The notation R is the reward function, which depends on the state–action pair and acts as an objective function. The notations b_0 and $\gamma \in (0,1)$ are the initial belief and discount factor.

At each time-step, a POMDP agent is at a state $s \in S$, takes an action $a \in A$, perceives an observation $o \in O$, receives a reward based on the reward function $R(s,a)$, and moves to the next state. Now due to uncertainty in the results of action and sensing, the agent never knows its exact state and therefore, estimates its state as a probability distribution, called belief. The solution to the POMDP problem is an optimal policy (denoted as π^*), which is a mapping $\pi^* : \mathbb{B} \to A$ from beliefs ($\mathbb{B}$ denotes the set of all beliefs, which is called the belief space) to actions that maximizes the expected total reward the robot receives, i.e., $V^*(b_0) = \max_{a \in A} \big(R(b,a) + \gamma \int_{o \in O} p(o|b,a) V^*(\tau(b,a,o))\, do\big)$, where $\tau(b,a,o)$ computes the updated belief estimate after the robot performs action $a \in A$ and perceived $o \in O$ from belief b, and is defined as:

$$b'(s') = \tau(b,a,o)(s') = \eta\, Z(s',a,o) \int_{s \in S} T(s,a,s') b(s) ds \tag{1}$$

For our motion planning problem, S, A, and O of the POMDP problem is the same as those of the robotic system (for simplicity, we use the same notation). The transition T represents the dynamics model f, while Z represents the sensing model h. The reward function represents the task' objective, for example, high reward for goal states and low negative reward for states that cause the robot to collide with the obstacles. The initial belief b_0 represents uncertainty on the starting state of the robot.

4 Statistical-distance-based Non-linearity Measure (SNM)

Intuitively, our proposed measure SNM is based on the total variation distance between the effect of performing an action and perceiving an observation under the true dynamics and sensing model, and the effect under the linearized dynamic and sensing model. The total variation distance $dist_{TV}$ between two probability functions θ and θ' over a measurable space Ω is defined as $dist_{TV}(\theta,\theta') = sup_{E \in \Omega} |\theta(E) - \theta'(E)|$. More formally, SNM is defined as:

Definition 1. *Let $P = \langle S,A,O,T,Z,R,b_0,\gamma \rangle$ be the POMDP model of the system and $\widehat{P} = \langle S,A,O,\widehat{T},\widehat{Z},R,b_0,\gamma \rangle$ be a linearization of P, where $\widehat{T}$ is a linearization of the transition function T and $\widehat{Z}$ is a linearization of the observation function Z of P, while all other components of P and $\widehat{P}$ are the same. Then, the SNM (denoted as Ψ) between P and $\widehat{P}$ is $\Psi(P,\widehat{P}) = \Psi_T(P,\widehat{P}) + \Psi_Z(P,\widehat{P})$, where*

$$\Psi_T(P,\widehat{P}) = \sup_{s \in S, a \in A} \Big| dist_{TV}(T(s,a,s'), \widehat{T}(s,a,s')) \Big| = \sup_{s,s' \in S, a \in A} \Big| T(s,a,s') - \widehat{T}(s,a,s') \Big|$$

$$\Psi_Z(P,\widehat{P}) = \sup_{s \in S, a \in A} \Big| dist_{TV}(Z(s,a,o), \widehat{Z}(s,a,o)) \Big| = \sup_{s \in S, a \in A, o \in O} \Big| Z(s,a,o) - \widehat{Z}(s,a,o) \Big|$$

Note that SNM can be applied as both a global and a local measure. For local measure, the supremum over the state s can be restricted to a subset of S, rather than the entire state space. Also, SNM is general enough for any approximation to the true dynamics and sensing model, which means that it can be applied to any type of linearization

and belief approximation techniques, including those that assume and those that do not assume Gaussian belief simplifications.

We want to use the measure $\Psi(P,\widehat{P})$ to bound the difference between the expected total reward received if the system were to run the optimal policy of the true model P and if it were to run the optimal policy of the linearized model $\widehat{P}$. Note that since our interest is in the actual reward received, the values of these policies are evaluated with respect to the original model P (we assume P is a faithful model of the system). More precisely, we want to show that:

Theorem 1. *Let π^* be the optimal policy of POMDP problem $P = \langle S,A,O,T,Z,R,b_0,\gamma\rangle$ and $\widehat{\pi}$ be the optimal policy of its linearized version $\widehat{P} = \langle S,A,O,\widehat{T},\widehat{Z},R,b_0,\gamma\rangle$, where $\widehat{T}$ is a linearization of the transition function T, $\widehat{Z}$ is a linearization of the observation function Z of P. Suppose the spaces S, A, and O for both models are the same and all represented as $[0,1]^n$, $[0,1]^d$, and $[0,1]^l$, respectively, with n, d, and l are the dimensions of the respective spaces. And, the reward function is Lipschitz continuous in S with Lipschitz constant K. Then,*

$$V_{\pi^*}(b_0) - V_{\widehat{\pi}}(b_0) \leq 2\gamma\left(\frac{R_{\max}}{(1-\gamma)^2} + \frac{KC\sqrt{n}}{1-\gamma}\right)\Psi(P,\widehat{P})$$

where $V_\pi(b) = R(b,\pi(b)) + \gamma\int_{o\in O} Z(b,a,o)V_\pi(\tau(b,a,o))do$ and $\tau(b,a,o)$ is the belief transition function as defined in (1) for the true model P. The notation C is a constant, defined as $C = \frac{1+\eta_1}{\eta_1\eta_2}$, where η_1 and η_2 are the normalization constants in the computation of $\tau(b,a,o)$ and its linearized version, respectively.

To prove this theorem, we first need to compute the upper bounds of two other difference functions. First is the total variation distance between the beliefs propagated under the true model P and under its linearized version $\widehat{P}$. More precisely,

Lemma 1. *Suppose $b' = \tau(b,a,o)$ and $\widehat{b'} = \widehat{\tau}(b,a,o)$ are the belief transition functions (as defined in (1)) for P and $\widehat{P}$, respectively. Then for any $a \in A$ and any $o \in O$,*

$$dist_{TV}(b',\widehat{b'}) \;=\; \sup_{s\in S}|b'(s) - \widehat{b'}(s)| \;\leq\; C\Psi(P,\widehat{P})$$

where $C = \frac{1+\eta_1}{\eta_1\eta_2}$, and η_1 and η_2 are the normalization constants in the computation of $\tau(b,a,o)$ and $\widehat{\tau}(b,a,o)$ respectively.

The second function is the difference between the observation function and its linearized version, given a belief and an action. Slightly abusing the notation Z, we use $Z(b,a,o)$ and $\widehat{Z}(b,a,o)$ to denote the conditional probability density (mass) function of perceiving observation $o \in O$ after action $a \in A$ is performed from belief b, under the model P and its linearized version $\widehat{P}$. We want to show that:

Lemma 2. *For any $b \in B$, $a \in A$, and $o \in O$, $Z(b,a,o) - \widehat{Z}(b,a,o) \;\leq\; \Psi(P,\widehat{P})$.*

In addition, we need to show that the optimal value function is Lipschitz continuous under the total variation distance, i.e.:

Lemma 3. *Let $P = \langle S,A,O,T,Z,R,b_0,\gamma\rangle$ be a POMDP problem where S is a metric space with dimension n and $|R(s,a) - R(s',a)| \leq K\,dist(s,s')$ for any $s \in S$ and $a \in A$. Then, $|V^*(b) - V^*(b')| \leq K\sqrt{n}\,dist_{TV}(b,b')$.*

The proofs for the above lemmas are available in Appendix A.

Now, we can show Theorem 1. For this purpose, we first divide the difference in the expected total reward into two components:

$$V_{\pi^*}(b_0) - V_{\widehat{\pi}}(b_0) = \left(V_{\pi^*}(b_0) - \widehat{V}(b_0)\right) + \left(\widehat{V}(b_0) - V_{\widehat{\pi}}(b_0)\right) \tag{2}$$

where $\widehat{V}(b_0)$ is the optimal value of the linearized model $\widehat{P}$ when the belief is at b_0.

We will start by computing an upper bound for the first component. Note that the value V_{π^*} is the same as the optimal value V^* of the POMDP problem P, and therefore we can compute the bound of the first component as:

$$\begin{aligned}
\left(V_{\pi^*}(b_0) - \widehat{V}(b_0)\right) &= \max_{a\in A}\left(R(b_0,a) + \gamma\int_{o\in O} Z(b_0,a,o)V^*(b_1)do\right) - \\
&\quad \max_{a\in A}\left(R(b_0,a) + \gamma\int_{o\in O} \widehat{Z}(b_0,a,o)\widehat{V}(\widehat{b_1})do\right) \\
&\leq \gamma\max_{a\in A}\left(\int_{o\in O} Z(b_0,a,o)V^*(b_1) - \widehat{Z}(b_0,a,o)\widehat{V}(\widehat{b_1})do\right) \\
&= \gamma\max_{a\in A}\left(\int_{o\in O}\left(Z(b_0,a,o)V^*(b_1) - Z(b_0,a,o)\widehat{V}(\widehat{b_1})\right) + \right. \\
&\quad \left.\left(Z(b_0,a,o)\widehat{V}(\widehat{b_1}) - \widehat{Z}(b_0,a,o)\widehat{V}(\widehat{b_1})\right)do\right) \\
&= \gamma\max_{a\in A}\left(\int_{o\in O}\left(Z(b_0,a,o)\left[V^*(b_1) - \widehat{V}(\widehat{b_1})\right] + \right.\right. \\
&\quad \left.\left.\left[Z(b_0,a,o) - \widehat{Z}(b_0,a,o)\right]\widehat{V}(\widehat{b_1})\right)do\right)
\end{aligned} \tag{3}$$

Replacing the difference between Z and $\widehat{Z}$ in the last term of (3) with the upper bound in Lemma 2 and assuming the volume of O is one, in addition to dividing the difference $[V^*(b_1) - \widehat{V}(\widehat{b_1})]$ into two components $(V^*(b_1) - \widehat{V}(b_1)) + (\widehat{V}(b_1) - \widehat{V}(\widehat{b_1}))$, allows us to rewrite (3) as:

$$\begin{aligned}
\left(V_{\pi^*}(b_0) - \widehat{V}(b_0)\right) \leq \gamma\,\Psi(P,\widehat{P})\frac{R_{\max}}{1-\gamma} + \gamma\max_{a\in A}\left(\int_{o\in O} Z(b_0,a,o)\left[V^*(b_1) - \widehat{V}(b_1)\right]do \right. \\
\left. + \int_{o\in O} Z(b_0,a,o)\left|\widehat{V}(b_1) - \widehat{V}(\widehat{b_1})\right|do\right)
\end{aligned} \tag{4}$$

We bound the last term of the right-hand-side of (4) using Lemma 3 and rewrite (4) as:

$$\begin{aligned}
\left(V_{\pi^*}(b_0) - \widehat{V}(b_0)\right) \leq \gamma\left(\Psi(P,\widehat{P})\frac{R_{\max}}{1-\gamma} + KC\sqrt{n}\,\Psi(P,\widehat{P})\right) + \\
\gamma\max_{a\in A}\left(\int_{o\in O} Z(b_0,a,o)\left[V^*(b_1) - \widehat{V}(b_1)\right]do\right)
\end{aligned}$$

Since V^* is equivalent to V_{π^*} for any beliefs, the last term in the above equation is essentially a recursion. Solving this recursion completes the upper-bound for the first component of (2), i.e.:

$$\left(V_{\pi^*}(b_0)-\widehat{V}(b_0)\right) \leq \gamma\left(\frac{R_{\max}}{(1-\gamma)^2}+\frac{KC\sqrt{n}}{1-\gamma}\right)\Psi(P,\widehat{P}) \tag{5}$$

Now, we compute the upper bound for the second component of (2) in a similar manner:

$$\begin{aligned}
\widehat{V}(b_0)-V_{\widehat{\pi}}(b_0) \leq & \left(R(b_0,\widehat{\pi}(b_0))+\gamma\int_{o\in O}\widehat{Z}(b_0,\widehat{\pi}(b_0),o)\widehat{V}(\widehat{b_1})do\right)- \\
& \left(R(b_0,\widehat{\pi}(b_0))+\gamma\int_{o\in O}Z(b_0,\widehat{\pi}(b_0),o)V_{\widehat{\pi}}(b_1)do\right) \\
= & \gamma\left(\int_{o\in O}\widehat{Z}(b_0,\widehat{\pi}(b_0),o)\widehat{V}(\widehat{b_1})-Z(b_0,\widehat{\pi}(b_0),o)V_{\widehat{\pi}}(b_1)do\right) \\
= & \gamma\left(\int_{o\in O}\left(\widehat{Z}(b_0,\widehat{\pi}(b_0),o)\left[\widehat{V}(\widehat{b_1})-V_{\widehat{\pi}}(b_1)\right]+\right.\right. \\
& \left.\left.\left[\widehat{Z}(b_0,\widehat{\pi}(b_0),o)-Z(b_0,\widehat{\pi}(b_0),o)\right]V_{\widehat{\pi}}(b_1)\right)do\right)
\end{aligned} \tag{6}$$

By dividing $\left[\widehat{V}(\widehat{b_1})-V_{\widehat{\pi}}(b_1)\right]$ into two components$\left(\widehat{V}(\widehat{b_1})-\widehat{V}(b_1)\right),\left(\widehat{V}(b_1)-V_{\widehat{\pi}}(b_1)\right)$ and using similar arguments we made in (3)–(5), we can bound the second component of (2) in the same way we did for its first component, i.e.:

$$\left(\widehat{V}(b_0)-V_{\widehat{\pi}}(b_0)\right) \leq \gamma\left(\frac{R_{\max}}{(1-\gamma)^2}+\frac{KC\sqrt{n}}{1-\gamma}\right)\Psi(P,\widehat{P}) \tag{7}$$

The sum of the right-hand-side of (5) and (7) yields the upper bound in Theorem 1.

The upper bound in Theorem 1 is relatively loose. However, the results in Section 6 indicate that this bound can be used as a sufficient condition to identify where linearization should and should not be applied.

5 SNM-Planner: An Application of SNM for Planning

SNM-Planner is an on-line planner that uses SNM as a heuristic to decide whether a general POMDP solver or a linearization-based motion planner should be used. The general solver used is Adaptive Belief Tree (ABT)[2], while the linearization-based method called Modified High Frequency Replanning (MHFR), which is an adaptation of HFR[3]. HFR is designed for chance-constraint POMDPs, i.e., it explicitly minimizes the collision probability, while MHFR is a POMDP solver where the objective is to maximize the expected total reward. During run-time, at each step, SNM-Planner approximates the local value of SNM around the current belief b. This value and a given threshold will then be used to decide whether to use ABT or MHFR to decide what action to take from b. An overview of the algorithm is in Algorithm 1.

5.1 Approximating SNM

Given the current belief b_i, SNM-Planner approximates the local value of SNM around b_i by approximating each component of SNM, i.e., Ψ_T and Ψ_Z, separately, using a simple Monte-Carlo approach. To approximate Ψ_T, SNM-Planner uses a Monte Carlo approach to construct a histogram representation of $T(s,a,s')$ in the support set of b_i. For

Algorithm 1 SNM-Planner (initial belief b_0, threshold μ, max planning time t, max time to approximate SNM t_m, #steps N, goal region $\mathbb{G}$)

$T_{ABT} \leftarrow InitializeABT(P)$
$T_{MHFR} \leftarrow InitializeMHFR(P)$
$t_p \leftarrow t - t_m$, $i \leftarrow 0$
while $s_i \notin G$ and collided = False and $i < N$ **do**
▷ s_i is the actual state at step-i. The system never knows s_i, but it knows if it is at a goal state.
    $\Psi \leftarrow approximatePsi(t_m, b_i)$
    if $\Psi < \mu$ **then**
        $a \leftarrow MHFR(T_{MHFR}, t_p, b_i)$
    else
        $a \leftarrow ABT(T_{ABT}, t_p, b_i)$
    end if
    executeAction(a)
    $o \leftarrow getObservation(b_i, a)$
    $b_{t+1} \leftarrow \tau(b_i, a, o)$ ▷ We use Sequential Importance Resampling [23]
    $i \leftarrow i + 1$
end while

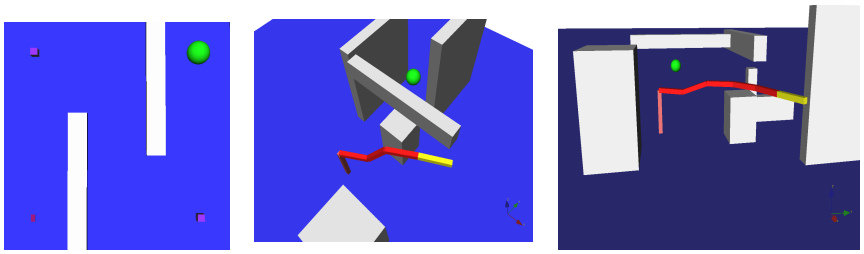
this purpose, SNM-Planner starts by sampling a set of state-action pairs (s,a), denoted as U, where each s is a state sampled from b_i and a is sampled uniformly at random from the action space A. For each pair (s,a), SNM-Planner samples a set of L possible next states according to $T(s,a,s')$ (denoted as $X_{(s,a)}$) and another set of L possible next states according to $\widehat{T}(s,a,s')$ (denoted as $\hat{X}_{(s,a)}$, where L is a given constant. It then constructs a histogram for each set of possible next states. Both histograms are constructed with respect to the same discretization of the state space S. Suppose K is the number of bins in this discretization, the approximate local value $\widehat{\Psi_T}(b_i)$ can be approximated as:

$$\widehat{\Psi_T}(b_i) \approx \max_{(s,a)\in U} \max_{k\in[1,K]} \frac{1}{L} \left| n^k_{s,a} - \hat{n}^k_{s,a} \right| \tag{8}$$

where $n^k_{s,a}$ is the number of elements in $X_{(s,a)}$ that lies in bin-k, while $\hat{n}^k_{s,a}$ is the number of elements in $\hat{X}_{(s,a)}$ that lies in bin-k.

To approximate Ψ_Z around b_i, the same procedure is performed. However, here, SNM-Planner samples the observation from $Z(s,a,o)$ and $\widehat{Z}(s,a,o)$. The histogram is then constructed by discretizing the observation space O.

At a first glance, the above method seems rather inefficient, due to the large number of histogram bins for high-dimensional state and observation spaces. However, three problem properties significantly reduce the computation cost. First, often a large portion of the histogram bins are empty and do not need to be considered, allowing for more efficient data structures, such as associative maps. Second, since SNM-Planner is a threshold-based method, as soon as the local approximation of SNM hits the threshold, the computation can be stopped. In our experiments we have seen that when the robot operates near obstacles where the local SNM is high, only a few state-action samples are needed to exceed the threshold. Last, calculating $\widehat{\Psi_T}^{s,a}(b_i)$ and $\widehat{\Psi_Z}(b_i)$ for different states and actions is trivially parallelizable.



(a) Maze (b) Factory-1 (c) 6DOF-scenario

Fig. 1: Test scenarios for the different robots. The objects colored grey are obstacles, while the green sphere is the goal region. (a) The car-like robot scenario. The purple square represents the beacons, while the red square at the bottom left represents the initial state. (b) The 4DOF-manipulator scenario. (c) The 6DOF-manipulator scenario. For (b-c), the robot is shown in red color with yellow end-effector.

6 Experimental Results

6.1 Experimental Setup

Our experiment is two-fold: To test SNM and to test the planner as proposed in Section 5. For our first objective, we compare SNM with a modified version of the measure of non-Gaussianity (MoNG) [1]. We use ABT as the general POMDP solver and MHFR as the linearization-based POMDP solver. Details of MoNG modifications, as well as ABT and MHFR are presented in Appendix B.

All algorithms are implemented in C++, while all experiments are conducted on Intel Xeon E5-2650 CPUs with 16GB RAM. For the parallel construction of the RRTs in MHFR, we utilize 8 CPU cores throughout the experiments. All parameters are set based on preliminary runs over the possible parameter space, the parameters that generate the best results are then chosen to generate the experimental results. For the comparison between SNM and MoNG, we use a car-like robot with 2^{nd} order control and a 4-DOFs manipulator with torque control. To test the proposed planner, we use these two scenarios plus a scenario involving a 6-DOFs manipulator with torque control. Details of these scenarios are as follows.

Car-like robot with 2^{nd} Order Control. A nonholonomic car-like robot of size (0.12×0.07) drives on a flat xy-plane inside a 3D environment populated by obstacles (Figure 1(a)). The robot must drive from a known start state to a position inside the goal region (marked as a green sphere) without colliding with any of the obstacles.

The state of the robot at time t is defined as a 4D vector $s_t = (x_t, y_t, \theta_t, \upsilon_t) \in \mathbb{R}^4$ where $x_t \in [-1,1]$ and $y_t \in [-1,1]$ are the position of the center of the robot on the xy-plane, $\theta_t \in [-3.14rad, 3.14rad]$ is the orientation, and $\upsilon_t \in [0, 0.2]$ is the linear velocity of the robot. The initial state of the robot is$(-0.7, -0.7, 1.57rad, 0)$, while goal region is centered at $(0.7, 0.7)$ with radius 0.1. The control input at time t, $a_t = (\alpha_t, \phi_t)$ is a 2D real vector consisting of the acceleration $\alpha \in [0,1]$ and the steering wheel angle $\phi \in [-1rad, 1rad]$. The robot's dynamics is subject to control noise $v_t = (\tilde{\alpha}_t, \tilde{\phi}_t)^T \sim N(0, \Sigma_v)$. The robot's transition model is

$$s_{t+1} = \left[x_t + \Delta t\, \upsilon cos\theta_t \ ; \ y_t + \Delta t\, \upsilon sin\theta_t \ ; \ \theta + \Delta t\ tan(\phi_t + \tilde{\phi})/0.11 \ ; \ \upsilon + \Delta t(\alpha_t + \tilde{\alpha})\right]$$

where Δt is the duration of a time step and the value 0.1 is the distance between the front and rear axles of the wheels.
The robot will enter a terminal state and receive a penalty of -500 if it hits an obstacle. It will also enter a terminal state, but with a reward of $1,000$ after reaching a goal region. All other actions incur a cost of -1. The robot localizes itself with the help of a velocity sensor mounted on the car and two beacons (marked with purple square in Figure 1(a)). Suppose the beacons are located at $(\hat{x}_1, \hat{y}_1)$ and $(\hat{x}_2, \hat{y}_2)$. In our experiment, the first beacon is at $(-0.7, 0.7)$ and the second beacon is at $(0.7, -0.7)$ Then, the signals the robot receives from these two beacons is a function of the distance to them, with additive Gaussian noise w_t. More formally, the robot's observation model is:

$$z_t = \left[1/((x_t - \hat{x}_1)^2 + (y_t - \hat{y}_1)^2 + 1); 1/((x_t - \hat{x}_2)^2 + (y_t - \hat{y}_2)^2 + 1); \upsilon_t\right] + w_t$$

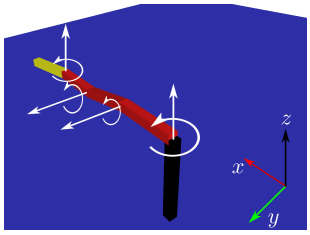


Fig. 2: The configuration of the 4DOFs-manipulator.

4-DOFs and 6-DOFs Manipulator with torque control. We describe these robotic systems in a general manner for a k-DOFs robot. This robot has k rotational joints, with limits at each of their joint angles and velocities, mounted on a static base. The manipulator operates in an environment populated by obstacles, and must move from the initial state to a state where the end-effector lies inside the goal region, without colliding with any of the obstacles. The environment scenarios for the 4-DOFs and 6-DOFs are in Figure 1(b) and Figure 1(c), respectively.

A state of the manipulator is defined as $s = (\theta, \dot{\theta}) \in \mathbb{R}^{2k}$, where $\theta \in [-3.14rad, 3.14rad]$ is the vector of joint angles and $\dot{\theta} \in [-3rad, 3rad]$ is the vector of rotational joint velocities. The rotational axes for the 4-DOFs manipulator are presented in Figure 2. For the 6-DOFs manipulator, the rotational axis of the first 4 joints are exactly the same as those of the 4-DOFs manipulator, and the additional two joints rotates around the Z axis. The mass of each link is 0.8kg. The control input $a \in A \subset \mathbb{R}^k$ is the joint torques. The torque limits for the 4-DOFs manipulator are $(\pm 20Nm/s, \pm 20Nm/s, \pm 10Nm/s, \pm 10Nm/s)$, while the torque limits for the 6-DOFs manipulator are $(\pm 20Nm/s, \pm 20Nm/s, \pm 20Nm/s, \pm 10Nm/s, \pm 10Nm/s, \pm 10Nm/s)$. The motion of the robot is disturbed by a k-dimensional error vector $v \sim N(0, \Sigma_v)$. The dynamics of the manipulator are modelled using the well-known Euler-Lagrangian formalism [24]. Note that although the error is Normally distributed, due to the non-linearity of the dynamics, the resulting belief estimate will generally not be Normally distributed. The initial state for both the 4DOFs and the 6DOFs manipulator is a state where all joint angles and joint velocities are zero. When the robot collides with an obstacle or with itself, it will move to a terminal state and receive a penalty of -500. When its end-effector reaches a goal region, the robot will move to a terminal state too, but it will receive a reward of 1000. All other actions incur a cost of -1. The robot is equipped with two types of sensors. The first sensor measures the position of the end-effector in the robot's workspace. The second sensor measures the joint velocities.

Empty environment				Maze scenario			
$e_T = e_Z$	**SNM**	**MoNG**	$\frac{\mathbf{V_{ABT}(b_0)-V_{MHFR}(b_0)}}{\mathbf{V_{ABT}(b_0)}}$	$e_T = e_Z$	**SNM**	**MoNG**	$\frac{\mathbf{V_{ABT}(b_0)-V_{MHFR}(b_0)}}{\mathbf{V_{ABT}(b_0)}}$
1.25	0.205	0.597	0.220	1.25	0.206	0.556	-0.660
2.50	0.200	0.611	0.262	2.50	0.213	0.675	-0.567
3.75	0.278	0.638	0.157	3.75	0.302	0.705	0.369
5.00	0.326	0.679	0.130	5.00	0.393	0.719	8.598

Table 1: The measure computed using SNM and MoNG, and the relative value difference between ABT and MHFR for the car-like robot in an empty environment, and in the maze scenario for increasing e_T and e_Z.

Suppose $g : \mathbb{R}^{2k} \mapsto \mathbb{R}^3$ is a function that maps the state of the robot to an end-effector position in the workspace and $w_t \sim N(0, \Sigma_w)$ is the error vector, then the observation model is defined as $z_t = [g(s_t), \dot{\theta}_t] + w_t$.

6.2 Testing SNM

In this set of experiments, we want to understand the performance of SNM compared to existing non-linearity measures for stochastic systems in various scenarios. In particular, we are interested in the effect that motion and sensing errors have and the effect that obstacles have on the effectiveness of SNM, compared to MoNG.

To this end, we perform experiments on the car-like robot and the 4-DOFs manipulator, with increasing the motion and sensing error of these robotic systems, operating in empty environments and environments populated by obstacles.

For experiments with increasing motion and sensing error, recall Section 6.1 that the control and sensing errors are drawn from zero-mean multivariate Gaussian distributions with covariance matrices Σ_v and diagonal entries $(\sigma_1, ..., \sigma_n)$. We then define relative control error (denoted as e_T) to be the percentage of the value range of the control inputs for each control dimension respectively. The square of the resulting values are then the diagonal entries of Σ_v. The observation error (denoted as e_Z) is defined in a similar fashion.

To investigate how SNM and MoNG perform as the control and sensing errors increase, we run experiments with multiple relative control and observation errors, ranging between 1.25% and 5.0%. To investigate the effect of obstacles to each measure, we ran each robotic system in an empty environment and in the environments as presented in Figure 1(a)-(b). For each scenario and each control-sensing error value (we set $e_T = e_Z$), we ran 100 simulation runs using ABT and MHFR, respectively. Since both ABT and MHFR are on-line planners, in each simulation run, each planner was allowed a planning time of 1s per planning step for the car-like robot, and 2s per planning step for the 4-DOFs manipulator. The average measures and value differences between ABT and MHFR are presented in Table 1 and Table 2.

The results indicate that in all scenarios, both SNM and MoNG are sensitive to an increase in the relative motion and sensing error. This increase generally resonates well with the increase in the difference between the average total discounted reward received if ABT were used and if MHFR were use, except for the case of the car-like robot operating in an empty environment. In this particular scenario, the relative difference between the general and the linearized solver decreases as the motion and sensing errors

Empty environment				Factory-1 scenario			
$e_T = e_Z$	**SNM**	**MoNG**	$\frac{\mathbf{V_{ABT}(b_0)-V_{MHFR}(b_0)}}{\mathbf{V_{ABT}(b_0)}}$	$e_T = e_Z$	**SNM**	**MoNG**	$\frac{\mathbf{V_{ABT}(b_0)-V_{MHFR}(b_0)}}{\mathbf{V_{ABT}(b_0)}}$
1.25	0.099	0.979	0.018	1.25	0.556	0.968	0.389
2.50	0.113	1.026	0.005	2.50	0.529	1.071	0.882
3.75	0.127	1.017	0.082	3.75	0.604	1.094	0.889
5.00	0.193	1.049	0.172	5.00	0.638	1.108	1.239

Table 2: The measure computed using SNM and MoNG, and the relative value difference between ABT and MHFR for the 4-DOFs manipulator in an empty environment, and in the factory-1 scenario for increasing e_T and e_Z.

increase. The reason is the performance of ABT decreases as the errors increase (similar as in the other three scenarios), but the performance of MHFR is almost unaffected. Figure 3(left) presents the plot of the average total reward of ABT and MHFR for the car-like robot operating in an empty environment. The performance of ABT decreases because as the motion and sensing errors increases, more particles are needed to represents the stochastic uncertainty well, which means if planning time per step does not increase, the quality of the generated strategy will decrease. The performance of MHFR is almost unaffected because in terms of computation time, MHFR is almost unaffected, it remains to use only the mean and covariance of the Gaussian distribution for planning. Furthermore, in this scenario, the penalty of making a wrong estimate will only be a longer route. Since the cost of a single action is -1 and due to the discount factor, the increase in the path length have an almost negligible effect on the total discounted reward.

Fig. 3: Mean total discounted reward when no obstacle is present.

Now, one may question why then the difference in value increases in the case of a 4-DOFs manipulator operating in an empty environment? As the plot in Figure 3(right) indicated, in this scenario, the performance of MHFR degrades as the relative motion and sensing errors increases. The reason is although the environment is empty, a 4-DOFs manipulator may have self-collision, and the increase in the motion and sensing errors causes the robot to be more susceptible to self-collision. Therefore, this scenario produces a similar trend to the test scenarios where obstacles are present (i.e., the maze and factory-1 scenarios).

In terms of sensitivity on the effect of obstacles to the effectiveness of linearization, both Table 1 and Table 2 indicate that SNM is more sensitive than MoNG. Overall, obstacles significantly increase the difference between the average total discounted reward if ABT were run and if MHFR were run. Similar to this trend, SNM shows significant increase in its non-linearity measure when obstacles are introduced. However, the measures computed using MoNG are unaffected by the introduction of obstacles.

6.3 Testing SNM-Planner

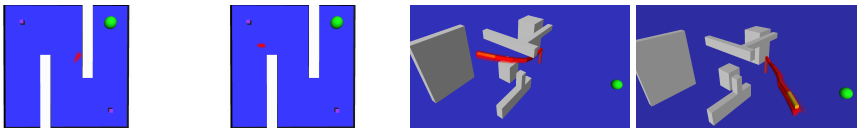
In this set of experiments, we want to test the performance of SNM-Planner. To this end, we tested our planner against the two component planners ABT and MHFR for three

different scenarios: The maze problem for the car-like robot, a 4-DOFs manipulator, and a 6-DOFs manipulator (as shown in Figure 1). We fixed both e_T and e_Z to 2.5% in these experiments. The SNM-threshold that is being used throughout these experiments is 0.3 for the car-like robot and 0.4 for both manipulator scenarios. The planning time that is being used for each algorithm is 1 sec per step for the car-like robot, 2 sec for the 4-DOFs manipulator and 7 sec for the 6-DOFs manipulator. Note that for SNM-Planner, the planning time per step consists of the time to approximate SNM and the planning time for the individual component planners. We allow a maximum of 20% of the total planning time per step for the approximation of SNM, while the other 80% is used for the component planners.

Planner	**Car-like robot**	**4-DOFs manipulator**	**6-DOFs manipulator**
ABT	128.59 $\pm$ 43.59	213.62 $\pm$ 64.68	737.23 $\pm$ 37.41
MHFR	141.39 $\pm$ 75.08	13.41 $\pm$ 116.76	265.39 $\pm$ 109.95
SNM-Planner	230.30 $\pm$ 70.56	382.62 $\pm$ 102.79	652.86 $\pm$ 69.48

Table 3: Mean total discounted reward +/- 95 % confidence interval over 100 simulation runs. The proportion of using ABT in the car-like robot, 4-DOFs and 6-DOFs manipulator scenarios are 29.53%, 36.13%, and 51.64% of the planning steps, respectively.

The results in Table 3 indicate that SNM-Planner can appropriately identify where to run linearization and where to not run linearization with a small enough cost, such that it can be used to generate motion strategies that are at least comparable to the suitable method.



(a) ABT is selected (b) MHFR is selected (c) ABT is selected (d) MHFR is selected.

Fig. 4: Typical situations for the car-like robot and the 6DOFs-manipulator robot for which the SNM is above and below the threshold

It is interesting to note that ABT is often selected when the robot operates in the vicinity of the obstacles. Figure 4 illustrates typical beliefs where ABT is selected (in the car-like robot and 6-DOFs manipulator scenarios). When the robot operates in the vicinity of an obstacle, SNM is usually larger than in free areas. In these situations, where careful planning is mandatory in order to avoid collisions, SNM-Planner prefers to use ABT, while in free areas where a more coarse planning would suffice, SNM-Planner prefers to use MHFR.

A critical aspect in SNM-Planner is how well can we approximate SNM in a limited time (i.e., 0.2s, 0.4s, and 1.4s for the car-like robot, 4-DOFs manipulator, and 6-DOFs manipulator, respectively). To understand this issue better, we tested the convergence rate of SNM in the car-like robot and the 4-DOFs manipulator scenario with various motion and sensing errors. For this purpose, we generate a trajectory for the maze and factory-1 scenario, and for each belief in the trajectory, we perform 100 independent Monte Carlo runs for different time limits to estimate the local value of SNM around the belief. The average of these estimates when we use the various time limits are presented in Figure 5.

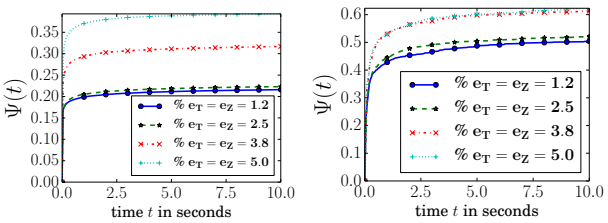


Fig. 5: Convergence to the true Ψ for the car-like robot (left) and 4DOFs-manipulator (right).

As expected, the results indicate that the size of the state, action, and observation space significantly influence the convergence rate. The car-like robot can converge to a good estimate faster than the 4-DOFs robot. It is interesting to note that the motion and sensing errors have little to no effect on the convergence of our method. These results also indicate that the estimate we use for testing SNM-Planner would have been a reasonable estimate, though not perfect, even though it takes very little time.

7 Summary and Future Work

This paper presents our preliminary work in identifying the suitability of linearization for motion planning under uncertainty. To this end, we present a general measure of non-linearity, called Statistical-distance-based Non-linearity Measure (SNM), which is based on the distance between the distributions that represent the system's motion–sensing model and its linearized version. Comparison studies with one of state-of-the-art methods for non-linearity measure indicate that SNM is more suitable in taking into account obstacles in measuring the effectiveness of linearization.

We also propose a simple on-line planner that uses a local estimate of SNM to select whether to use a general POMDP solver or a linearization-based solver for robot motion planning under uncertainty. Experimental results indicate that our simple planner can appropriately decide where linearization should be used and generates motion strategies that are comparable or better than each of the component planner.

Future work abounds. For instance, the question for a better measure remains. Total variation distance relies on computing a maximization, which is often difficult to estimate. Statistical distance function that relies on expectation exists and can be computed faster. How suitable are these functions as a non-linearity measure? Furthermore, our upper bound result is relatively loose and can only be applied as a sufficient condition to identify if linearization will perform well. It would be useful to find a tighter bound that remains general enough for the various linearization and distribution approximation methods in robotics.

References

1. Duník, J., Straka, O., Šimandl, M.: Nonlinearity and non-gaussianity measures for stochastic dynamic systems. In: Information Fusion (FUSION), IEEE (2013) 204–211
2. Kurniawati, H., Yadav, V.: An online POMDP solver for uncertainty planning in dynamic environment. In: ISRR. (2013)
3. Sun, W., Patil, S., Alterovitz, R.: High-frequency replanning under uncertainty using parallel sampling-based motion planning. IEEE Transactions on Robotics **31**(1) (2015) 104–116
4. Canny, J., Reif, J.: New lower bound techniques for robot motion planning problems. In: Foundations of Computer Science, 1987., 28th Annual Symposium on, IEEE (1987) 49–60
5. Natarajan, B.: The complexity of fine motion planning. The International journal of robotics research **7**(2) (1988) 36–42

6. Kaelbling, L., Littman, M., Cassandra, A.: Planning and acting in partially observable stochastic domains. AI **101** (1998) 99–134
7. Drake, A.W.: Observation of a Markov process through a noisy channel. PhD thesis, Massachusetts Institute of Technology (1962)
8. Horowitz, M., Burdick, J.: Interactive Non-Prehensile Manipulation for Grasping Via POMDPs. In: ICRA. (2013)
9. Temizer, S., Kochenderfer, M., Kaelbling, L., Lozano-Pérez, T., Kuchar, J.: Unmanned aircraft collision avoidance using partially observable markov decision processes. Project Report ATC-356, MIT Lincoln Laboratory, Advanced Concepts Program, Lexington, Massachusetts, USA (September 2009)
10. Silver, D., Veness, J.: Monte-Carlo Planning in Large POMDPs. In: NIPS. (2010)
11. Somani, A., Ye, N., Hsu, D., Lee, W.S.: DESPOT: Online POMDP planning with regularization. In: NIPS. (2013) 1772–1780
12. Seiler, K., Kurniawati, H., Singh, S.: An online and approximate solver for pomdps with continuous action space. In: ICRA. (2015)
13. Agha-Mohammadi, A.A., Chakravorty, S., Amato, N.M.: Firm: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements. IJRR (2013)
14. Berg, J., Abbeel, P., Goldberg, K.: LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information. In: RSS. (2010)
15. Berg, J., Wilkie, D., Guy, S., Niethammer, M., Manocha, D.: LQG-Obstacles: Feedback Control with Collision Avoidance for Mobile Robots with Motion and Sensing Uncertainty. In: ICRA. (2012)
16. Prentice, S., Roy, N.: The belief roadmap: Efficient planning in linear pomdps by factoring the covariance. In: Robotics Research. Springer (2010) 293–305
17. Li, X.R.: Measure of nonlinearity for stochastic systems. In: Information Fusion (FUSION), 2012 15th International Conference on, IEEE (2012) 1073–1080
18. Bates, D.M., Watts, D.G.: Relative curvature measures of nonlinearity. Journal of the Royal Statistical Society. Series B (Methodological) (1980) 1–25
19. Beale, E.: Confidence regions in non-linear estimation. Journal of the Royal Statistical Society. Series B (Methodological) (1960) 41–88
20. Emancipator, K., Kroll, M.H.: A quantitative measure of nonlinearity. Clinical chemistry **39**(5) (1993) 766–772
21. Mastin, A., Jaillet, P.: Loss bounds for uncertain transition probabilities in markov decision processes. In: CDC, IEEE (2012) 6708–6715
22. Müller, A.: How does the value function of a markov decision process depend on the transition probabilities? Mathematics of Operations Research **22**(4) (1997) 872–885
23. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on signal processing **50**(2) (2002) 174–188
24. Spong, M.W., Hutchinson, S., Vidyasagar, M.: Robot Modeling and Control. Volume 3. Wiley New York (2006)
25. Kurniawati, H., Patrikalakis, N.: Point-Based Policy Transformation: Adapting Policy to Changing POMDP Models. In: WAFR. (2012)
26. Gibbs, A.L., Su, F.E.: On choosing and bounding probability metrics. International statistical review **70**(3) (2002) 419–435
27. Lavalle, S.M., Kuffner Jr, J.J.: Rapidly-exploring random trees: Progress and prospects. In: Algorithmic and Computational Robotics: New Directions, Citeseer (2000)

Appendix A Proofs of Lemma 1 — Lemma 3

To prove Lemma 1 and Lemma 2, we need a notion of difference between the transition function T of the true POMDP model P and its linearized version $\widehat{T}$ in $\widehat{P}$. To this end, we define the difference as: $D_S(s,a,s') = T(s,a,s') - \widehat{T}(s,a,s')$. We also need a similar notion of difference for the observation function. For this purpose, we define this difference to be: $D_O(s,a,o) = Z(s,a,o) - \widehat{Z}(s,a,o)$. Note that these two notion of differences are upper-bounded by the components of SNM, i.e., $D_S(s,a,s') \leq \Psi_T(P,\widehat{P})$ and $D_O(s,a,o) \leq \Psi_Z(P,\widehat{P})$ for any $s,s' \in S$, $a \in A$, and $o \in O$.

A.1 Proof of Lemma 1

Without loss of generality, let's first compute $|b'(s') - \widehat{b}'(s')|$ for state $s' \in S$ as:

$$\left|b'(s') - \widehat{b}'(s')\right| = \frac{1}{\eta_1}\left(Z(s',a,o)\int_{s\in S} T(s,a,s')b(s)ds\right) - \frac{1}{\eta_2}\left(\widehat{Z}(s',a,o)\int_{s\in S}\widehat{T}(s,a,s')b(s)ds\right)$$

Replacing the linearized function $\widehat{T}$ and $\widehat{Z}$ with the notion of difference D_S and D_O allows us to expand the right-hand-side of the above equation into:

$$= \frac{1}{\eta_1\eta_2}\left(\eta_2\left(Z(s',a,o)\int_{s\in S} T(s,a,s')b(s)ds\right) - \eta_1\left(\left(Z(s',a,o) - D_O(s',a,o)\right)\int_{s\in S}\left(T(s,a,s') - D_S(s,a,s')\right)b(s)ds\right)\right)$$

Manipulating the algebra allows us to expand the above terms further into:

$$\begin{aligned}\left|b'(s') - \widehat{b}'(s')\right| &= \frac{1}{\eta_1\eta_2}\Bigg((\eta_2-\eta_1)\int_{s\in S} b(s)T(s,a,s')Z(s',a,o)ds \\ &\quad + \eta_1\int_{s\in S} b(s)\left(T(s,a,s') - D_S(s,a,s')\right)D_O(s',a,o)ds + \eta_1\int_{s\in S} b(s)Z(s',a,o)D_S(s,a,s')ds\Bigg) \\ &= \frac{1}{\eta_1\eta_2}\Bigg((\eta_2-\eta_1)\int_{s\in S} b(s)T(s,a,s')Z(s',a,o)ds \\ &\quad + \eta_1\int_{s\in S} b(s)\widehat{T}(s,a,s')D_O(s',a,o)ds + \eta_1\int_{s\in S} b(s)Z(s',a,o)D_S(s,a,s')ds\Bigg)\end{aligned}$$

Using the upper bound of the differences, we can bound the above equation as:

$$\left|b'(s') - \widehat{b}'(s')\right| \leq \frac{1}{\eta_1\eta_2}\left((\eta_2-\eta_1) + \eta_1\Psi_T(P,\widehat{P}) + \eta_1\Psi_Z(P,\widehat{P})\right) \tag{9}$$

Let's now compute the upper bound for the first term of the right-hand-side of the above equation:

$$\begin{aligned}\eta_2 - \eta_1 &= \int_{s'\in S}\int_{s\in S}\left(\widehat{Z}(s',a,o)\widehat{T}(s,a,s') - Z(s',a,o)T(s,a,s')\right)b(s)dsds' \\ &= \int_{s'\in S}\int_{s\in S}\left(\left(Z(s',a,o) - D_O(s',a,o)\right)\left(T(s,a,s') - D_S(s,a,s')\right) - Z(s',a,o)T(s,a,s')\right)b(s)dsds' \\ &= \int_{s'\in S}\int_{s\in S}\left(\widehat{T}(s,a,s')D_O(s',a,o) - Z(s',a,o)D_S(s,a,s')\right)b(s)dsds' \\ &\leq \Psi_Z(P,\widehat{P}) + \Psi_T(P,\widehat{P})\end{aligned} \tag{10}$$

Substituting the term $(\eta_2 - \eta_1)$ in (9) with its upper bound as derived in (10) will yield the upper bound in this lemma. □.

A.2 Proof of Lemma 2

Given an observation $o \in O$, we can compute the difference in the observation function of the true and linearized models after action $a \in A$ is performed from belief b as:

$$Z(b,a,o) - \widehat{Z}(b,a,o) = \int_{s' \in S} \int_{s \in S} b(s) \left(Z(s',a,o) T(s,a,s') - \widehat{Z}(s',a,o) \widehat{T}(s,a,s') \right) ds ds'$$

Replacing the linearized transition and observation function with the difference equation D_S and D_O allows us to expand the right-hand-side of the above equation into:

$$\begin{aligned} Z(b,a,o) - \widehat{Z}(b,a,o) &= \int_{s' \in S} \int_{s \in S} b(s) \left(Z(s',a,o) D_S(s,a,s') + T(s,a,s') D_O(s',a,o) - D_S(s,a,s') D_O(s',a,o) \right) ds ds' \\ &= \int_{s' \in S} \int_{s \in S} b(s) \left(Z(s',a,o) D_S(s,a,s') + \left(T(s,a,s') - D_S(s,a,s') \right) D_O(s',a,o) \right) ds ds' \\ &= \int_{s' \in S} \int_{s \in S} b(s) \left(Z(s',a,o) D_S(s,a,s') + \widehat{T}(s,a,s') D_O(s',a,o) \right) ds ds' \end{aligned}$$

Replacing D_S and D_O with their upper bounds and integrating the probabilities out will yield the upper bound we want to prove, i.e.: $Z(b,a,o) - \widehat{Z}(b,a,o) \leq \Psi_Z(P,\widehat{P}) + \Psi_T(P,\widehat{P})$. □.

A.3 Proof of Lemma 3

We know that if $|R(s,a) - R(s',a)| \leq K\, dist(s,s')$, then $|V^*(b) - V^*(b')| \leq K dist_W(b,b')$, where $dist_W(b,b')$ is the Wasserstein distance between beliefs b and b' [25]. Using Lemma 1, we know that $dist_{TV}(b_1, \widehat{b_1}) \leq C\Psi(P,\widehat{P})$. Since the Wasserstein distance between any two distributions is upper bounded by a linear function of the total variation distance, i.e., $dist_W(b,b') \leq \sqrt{n} \cdot dist_{TV}(b,b')$[26]. Then, $\left| \widehat{V}(b_1) - \widehat{V}(\widehat{b_1}) \leq KC\sqrt{n}\, \Psi(P,\widehat{P}) \right|$, which is Lemma 3. □.

Appendix B Implementation Details of Relevant Algorithms

B.1 The Comparator Non-Linearity Measures

The measure of Non-Gaussianity (MonG) proposed in [1] is based on the negentropy between the PDF of a random variable and its Gaussian approximation. Consider a n-dimensional random variable x distributed according to PDF $p(x)$. Furthermore, let $\widehat{x}$ be a Gaussian approximation of x with PDF $\widehat{p}$, such that $\widehat{x} \sim N(\mu, \Sigma_x)$, where μ and Σ_x are the first two moments of x. The negentropy between p and $\widehat{p}$ (denoted as $J(p,\widehat{p})$) is then defined as

$$\begin{aligned} J(p,\widehat{p}) &= H(\widehat{p}) - H(p) \\ H(\widehat{p}) &= \frac{1}{2} ln\left[(2\pi e)^n \left| det(\Sigma_x) \right| \right] \\ H(p) &= -\int p(x) \ln p(x) dx \end{aligned} \tag{11}$$

where $H(p), H(\widehat{p})$ is the differential entropy of p and $\widehat{p}$ respectively. A (multivariate) normal distribution has the largest differential entropy amongst all distributions with equal first two moments, therefore $J(p,\widehat{p})$ is always non-negative. In practice, since the PDF $p(x)$ is not known exactly in all but the simplest cases, $H(p)$ has to be approximated.

In [1] this measure has originally been used to assess the nonlinearity of passive systems. Therefore, in order to achieve comparability with SNM, we need to extend the Non-Gaussian

measure to general active stochastic systems of the form $s_{t+1} = f(s_t, a_t, v_t)$. We do this by evaluating the non-Gaussianity of distribution that follows from the transition function $T(s,a,s')$ given state s and action a. In particular for a given s and a, we can find a Gaussian approximation of $T(s,a,s')$ (denoted by $T_G(s,a,s')$) by calculating the first two moments of the distribution that follows from $T(s,a,s')$.

Using this Gaussian approximation, we define the Measure of Non-Gaussianity as

$$MoNG(T,T_G) = \sup_{s\in S, a\in A} \left[H(T(s,a,s')) - H(T_G(s,a,s'))\right] \tag{12}$$

In order to approximate $H(T(s,a,s'))$, we employ a similar Monte-Carlo based approach as discussed in Section 5.1.

Note that throughout the experiments conducted in this paper, we use observation functions with additive Gaussian noise, which results in the conditional observation distribution that follows from $Z(s,a,o)$ being a translated Gaussian distribution, for which MoNG evaluates to 0 for any state and action. Hence we only evaluate MoNG for the nonlinear transition functions.

B.2 Adaptive Belief Tree (ABT)

ABT is an on-line and anytime general POMDP solver that updates (rather than recomputes) its policy at each planning step. Given the current belief b_t, ABT constructs and maintains a belief-tree by sampling episodes sequences of state–action–observation–reward tuples, starting from b_t using a generative model. Each node in the belief tree represents a belief, while an edge from belief b to b' means that there is an action $a \in A$ and an observation $o \in O$, such that $b' = \tau(b,a,o)$. Details of the method is in [2].

B.3 Modified High Frequency Replanning (MHFR)

The main difference between HFR and MHFR is that HFR is designed for chance constraint POMDP, i.e., it explicitly minimizes the collision probability, while MHFR is a POMDP solver, whose objective is to maximize the expected total reward. Similar to HFR, MHFR computes the solution on-line and approximates b_t by a Normal distribution $N(\bar{s}, \Sigma)$.

To decide the best action to perform from the initial belief $b_0 = N(\overline{s_0}, \Sigma_0)$, MHFR computes multiple trajectories from $\overline{s_0}$ to a goal state in parallel, using RRT [27]. To decide which trajectory to follow, MHFR computes the expected total discounted reward of each trajectory, assuming the most likely observation is perceived at every step and tracking the changes in the system's belief using KF. Once the best sampled trajectory is found, the first action $a \in A$ is performed, an observation $o \in O$ is perceived, and an EKF is applied to compute the current belief $b_1 = N(\overline{s_1}, \Sigma_1)$ given the previous belief, the action performed, and the actual observation that the system perceived. Now, to compute the action from b_1, MHFR computes multiple trajectories from s_1 to a goal state using RRT, and uses LQG to adjust the best trajectory from b_0. For each of these trajectories (the newly sampled trajectories and the adjusted one), MHFR computes the expected total discounted reward, so as to find the best trajectory to use from b_1. Once the best trajectory is found, the same procedure as above is performed, and the process repeats.