Learned Sensitivity Metrics for Robust and Accurate Motion Planning
Simon Wasiela, Marco Cognetti, Paolo Robuffo Giordano, Juan Cortés, Thierry Simeon

To cite this version:
Simon Wasiela, Marco Cognetti, Paolo Robuffo Giordano, Juan Cortés, Thierry Simeon. Learned Sensitivity Metrics for Robust and Accurate Motion Planning. 2023. hal-04027078

HAL Id: hal-04027078
https://hal.laas.fr/hal-04027078
Preprint submitted on 13 Mar 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Learned Sensitivity Metrics for Robust and Accurate Motion Planning

Simon Wasiela¹, Marco Cognetti¹,², Paolo Robuffo Giordano³, Juan Cortés¹ and Thierry Siméon¹

Abstract—This paper addresses the problem of computing robust trajectories against uncertainties in the robot model. Based on the notion of closed-loop state sensitivity introduced in [1], [2] for identifying deviations of the closed-loop trajectories of any robot/controller pair against variations of uncertain parameters in the robot model, uncertainty tubes can be derived for bounded parameter variations [3]. Such tubes were integrated within a motion planner named SAMP [4] to produce robust global plans, emphasizing the generation of low sensitivity trajectories. However, a bottleneck of this method remains the very high computational cost of the uncertainty tubes. In this paper, we mitigate this problem by proposing a novel framework that first incorporates a Long Short-Term Memory (LSTM) neural network (NN) to provide fast and accurate uncertainty tubes estimation, and then reduces tracking errors at given points on the trajectory. We validate our framework on a full 3D-quadrotor UAV model performing a ‘ring-recovering’ task requiring high accuracy. Our results show the computational gain of the LSTM-based robust planning and that the subsequent optimization stage significantly improves the accuracy of the trajectory execution.

I. INTRODUCTION

Nowadays, motion planners can generate feasible and globally-optimal paths for high-dimensional systems and complex constraints/environments (e.g. see the recent survey [6]). However, most planners do not consider the unavoidable presence of uncertainties in the robot/environment model, and of a feedback action that will track the planned trajectories in presence of these uncertainties. This can often lead to a poor robustness and potential loss of optimality at runtime.

Efforts have been made to merge control and planning together. For example, the well known Model Predictive Control (MPC) technique [7] provides a “less local” control strategy. This technique consists in re-planning a local optimal trajectory taking into account a state feedback and a prediction of the evolution of the system state over a finite horizon. Nevertheless, MPC remains local in the vicinity of the reference trajectory. On the other hand, the global control-aware motion planner proposed in [8] relies on some coupling between planning and control, but uncertainties are not explicitly considered.

Other approaches based on contraction theory [9]–[13] or the so-called ‘feedback motion planning’ [14], [15] synthesize or rely on a specific robust controller that ensures that the trajectory remains within some uncertainty tubes, which can then be used in global planners as RRT [16]. These methods, however, remain limited to a specific form of control strategy and are often applicable only to specific class of systems for which the controller derivation is feasible in practice.

To overcome some of these problems, the sensitivity-aware motion planner (SAMP) [4] exploits the derivation of the uncertainty tubes in [3] (based on the so-called closed-loop sensitivity matrix [1], [2]) to generate globally robust trajectories for any controller and any robot dynamics. However, this planner suffers from the high computational cost of calculating the uncertainty tubes.

The contribution presented in this paper is twofold. We first propose a more efficient sampling-based motion planner (SBMP) relying on a Long Short-Term Memory (LSTM) [5] network that quickly and accurately estimates the uncertainty tubes and control input profiles along trajectories. In addition, the proposed framework locally optimizes the planned trajectory together with the controller gains in order to maximize the accuracy at a set of waypoints. Simulation results show the higher computational efficiency of the planning framework. Also, we tested our approach for the scenario shown in Fig.1 where a 3D-quadrotor with uncertain parameters has to perform an in-flight ring recovery task. The results confirm a better accuracy and robustness of the plan.

The paper is organized as follows. Sect. II first provides a reminder of the closed-loop sensitivity notions, and then describes our machine learning approach. Sect. III presents a framework to generate robust and accurate trajectories, and explains the incorporation of the LSTM-based neural network into a global planner. The dynamics of a 3D

Fig. 1: Scenario with a drone that has to grasp a ring. (a) the uncertainty associated with its end-effector position is too large and the execution fails. (b) with a lower uncertainty the robot is able to successfully accomplish the task.
quadrotor, the application of the learning method to the latter, and the planning scenario are described in Sect. IV. Simulations results are provided in Sect. V. Finally we draw some conclusions and future directions in Sect. VI.

II. LEARNING SENSITIVITY METRICS

A. Closed-loop sensitivity: Reminder

The notion of closed-loop state sensitivity [1], [2] quantifies deviations of the closed-loop trajectory of any robot/controller pair against variations of uncertain parameters in the robot model. Consider a generic robot dynamics

$$\dot{q} = f(q, u, p), \quad q(t_0) = q_0,$$  \hspace{1cm} (1)

where $q \in \mathbb{R}^n_q$ is the state vector, $u \in \mathbb{R}^n_u$ the input vector, and $p \in \mathbb{R}^n_p$ the vector containing (possibly uncertain) model parameters. Also assume presence of a tracking controller of any form to track a reference trajectory $r_d(a, t)$ parameterized by vector $a$ s.t.,

$$\begin{cases}
\dot{\xi} = g(\xi, q, a, p_c, k_c, t), & \xi(t_0) = \xi_0 \\
u = h(\xi, q, a, p_c, k_c, t),
\end{cases}$$  \hspace{1cm} (2)

where $\xi \in \mathbb{R}^n_\xi$ are the internal states of the controller (e.g., an integral action), $k_c \in \mathbb{R}^n_k$ the controller gains and $p_c \in \mathbb{R}^n_p$ the vector of nominal robot parameters used in the control loop.

In order to quantify how sensible the system is w.r.t. parameter variations, let the state sensitivity matrix $\Pi$ be defined as (see [1])

$$\Pi(t) = \frac{\partial q(t)}{\partial p}|_{p=p_{c,nom}} \in \mathbb{R}^{n_q \times n_p},$$  \hspace{1cm} (3)

and the input sensitivity matrix as (see [2])

$$\Theta(t) = \frac{\partial u(t)}{\partial p}|_{p=p_{c,nom}} \in \mathbb{R}^{n_u \times n_p}.$$  \hspace{1cm} (4)

These two matrices quantify, respectively, how sensible the states $q(t)$ and the inputs $u(t)$ are w.r.t. variations of $p$ (w.r.t. the ‘nominal’ $p_{c,nom}$) for the closed-loop system (1–2). They do not have in general a closed-form expression, but it is possible to obtain them in practice by numerical integration of a first-order differential equation along a planned trajectory.

The sensitivity framework has been extended in [3] in the case where each parameter $p_i$ is assumed to vary within a range $\delta p_i$ centered on the parameter nominal value $p_{c,i}$ s.t. $p_i \in [p_{c,i} - \delta p_i, p_{c,i} + \delta p_i]$. In these conditions, the so-called uncertainty tubes can be computed as the envelope of the perturbed state/input trajectories obtained when the parameters vary in the given range. In order to compute these tubes, let $W = \text{diag}(\delta p_i^2)$ be a weighting matrix and

$$\Delta p^T W^{-1} \Delta p = 1,$$  \hspace{1cm} (5)

be an ellipsoid in parameter space centered at $p_{c}$ and semi-axis $\delta p_i$, with $\Delta p = p - p_{c}$. Assuming small variations of the parameters, one has $\Delta q \approx \Pi(t) \Delta p$, and it is possible to obtain – see [3] for details – the corresponding ellipsoids in the state and the control spaces as

$$\Delta q^T (\Pi W \Pi^T)^{-1} \Delta q = 1,$$  \hspace{1cm} (6)

$$\Delta u^T (\Theta W \Theta^T)^{-1} \Delta u = 1,$$  \hspace{1cm} (7)

where $\Delta q = q - q_{c,nom}, \Delta u = u - u_{c,nom}$, while $q_{c,nom}$ (resp. $u_{c,nom}$) is the state (resp. input) evolution of the closed-loop system in eqs. (1-2) in the nominal case $(p = p_{c,nom})$. These ellipsoids represent the sets where the states and the inputs can vary in case the parameters vary in the specified ranges. Following the procedure given in [3], it is further possible to compute the radius $r_i(t)$ along the $i$-th component of the state that bounds its evolution over time. In order words,

$$q_{c,nom}(t) - r_i(t) \leq q_i(t) \leq q_{c,nom}(t) + r_i(t).$$  \hspace{1cm} (8)

A similar derivation can also be done for the input ellipsoid. Such radii were used in [4] to perform robust collision checking within a motion planning framework. However, their computations rely on the knowledge of $\Pi(t)$ and $\Theta(t)$ that have been shown to be computationally expensive in the context of sampling-based motion planning, resulting in high planning times even for simple problems.

B. LSTM-based learning method

To leverage this computational issue, we propose a learning-based method to quickly and accurately estimate the tube radii in eq. (8) and the input profiles in eq. (2).

LSTM [5] is a recurrent neural network (NN) that has feedback connections encoding past events thanks to the so-called hidden and cell states. LSTM NN has already been used in recent years for motion planning applications, e.g., to quickly predict the surrounding state of the robot [17], [18], or to predict trajectories in sampling based algorithms [19]. Since LSTM NNs work very well in processing sequences of data – in particular time series – they perfectly fit with our framework, since the robot trajectory can be discretized.
in some points (hereafter referred to as state) associated with some time instants, that will be treated as input sequences by the LSTM NN.

A simplified architecture of our NN is presented in Fig. 2, where the tuple \((h_0, c_0)\) represents the initial hidden and cell states, respectively. The input sequence consists of a series of states, denoted with \(q_k\), evaluated along a trajectory at a given time step. The output is a sequence consisting of the inputs values \(U^k\) and of \(R_t^k\) and \(R_u^k\), that represent the radii of the uncertainty tubes, estimated at the \(k\)-th state of the trajectory, along the desired direction of the state and of the input spaces respectively. Finally, the tuple \((h_{end}, c_{end})\) corresponds to the hidden and the cell states at the last point of our sequence (i.e., the last state of our trajectory). The importance of such initial and final tuples is discussed in Sect. III-B. Training and prediction results are presented in Sect. V, as well as a discussion on the limitations.

III. ROBUST AND ACCURATE MOTION PLANNING VIA LEARNED SENSITIVITY METRICS

We first present the overall planning framework proposed to generate both robust and accurate trajectories in Sect. III-A. The two components of this framework are described next: Sect. III-B shows how to integrate the sensitivity learning method within a sampling-based motion planner, Sect. III-C then explains the accuracy optimization stage.

A. Robust and accurate planning framework (RA-SAMP)

In this section, we present RA-SAMP, a planning framework that generates robust and accurate trajectories. It works in two stages: (i) first, it generates a robust trajectory based on a Robust Sensitivity-Aware Motion Planner (R-SAMP) integrating the LSTM-based computation of the uncertainty tubes; (ii) second, it optimizes the accuracy at some given points of this trajectory by minimizing the size of the uncertainty tubes at these locations. The motivation behind the second stage is to improve the accuracy of the planned robust trajectory for tasks – e.g., pick-and-place or insertion tasks – where the precision is important only at specific designed locations, as for picking the ring in Fig. 1.

We present RA-SAMP in its most general form in Alg.1. It takes as inputs the list of waypoints \(\text{list}_\text{wpt} = (w_0, \ldots, w_n)\) for which the accuracy should be optimized, and the values of the nominal controller gains.

**Algorithm 1 RA-SAMP** \([\text{list}_\text{wpt}, \text{gains}_\text{init}]\)

1: \(\{\pi_{\text{tot}}, \text{gains}_{\text{best}}\} \leftarrow \emptyset, \text{gains}_\text{init}\);
2: for \(i = 1; i < \text{len}(\text{list}_\text{wpt}); i = i + 1\) do
3: \(\pi_i \leftarrow \text{R-SAMP}(\text{list}_\text{wpt}(i - 1), \text{list}_\text{wpt}(i))\);
4: \(\pi_{\text{tot}} \leftarrow \pi_{\text{tot}} + \pi_i\);
5: end for
6: \(\{\pi_{\text{tot}}, \text{gains}_{\text{best}}\} \leftarrow \text{A-Optim}(\pi_{\text{tot}}, \text{gains}_{\text{best}})\);
7: return \(\{\pi_{\text{tot}}, \text{gains}_{\text{best}}\}\);

The first step of the algorithm consists in generating robust trajectories between successive waypoints in the list (line 3 of Alg. 1) by means of a SBMP framework called R-SAMP that utilizes our learning approach. These trajectories are concatenated into a global one, interconnecting all the waypoints (lines 2-5 in Alg. 1).

Then, the trajectory from R-SAMP is locally modified by A-Optim (line 6 in Alg. 1), an algorithm whose aim is to optimize the accuracy at given waypoints along the input trajectory. In particular, this algorithm samples, at each iteration, both the trajectory from R-SAMP and the controller’s gains, modifying the former in order to minimize the uncertainty at the waypoints. The outputs of the algorithm are then: (i) a robust trajectory where the accuracy is maximized at the desired waypoints; (ii) the optimal controller gains.

Note that, even if R-SAMP benefits from the neural network, A-Optim cannot employ it because the controller gains are different from those used for training the network.

B. Robust sensitivity-aware motion planning (R-SAMP)

In this section, we describe how the learned uncertainty tubes can be incorporated in any tree-based SBMP in order to generate a robust sensitivity-aware motion planner (R-SAMP). As discussed before, one issue for computing the sensitivity metrics along a given trajectory comes from the possibly high computational time of the numerical integration of the dynamics for obtaining \(\Pi(t)\) and \(\Theta(t)\). This time might span from tens of milliseconds to several seconds – depending on the trajectory duration – and it sensibly increases the planning time for a SBMP. Moreover, an additional problem – that is specific when integrating the sensitivity machinery within a SBMP – comes from the fact that, for computing the sensitivity for a given node of a SBMP, the overall path for arriving at that node is needed.

However, this is not a problem for our framework thanks to the LSTM network, that naturally encodes this information within the hidden and the cell states. To this aim, these states are saved for each node of the tree built by the SBMP. Then, when expanding a node, the starting hidden and cell states (as for the tuple \((h_0, c_0)\) of Fig.2) are already available from previous iterations, and the final ones \((h_{end}, c_{end})\) are embedded within the node generated by the expansion. This last tuple can then be reused in future extensions as the initial condition tuple \((h_0, c_0)\). Note that such mechanism of using the tuples \((h, c)\) limits its validity to tree-based planners (e.g., [20]–[24]), where each node has a single parent. Therefore, planners whose nodes may have multiple parents (e.g., PRM [25]) cannot employ such mechanism.

The LSTM network is used by the tree-based planner inside the function RobustCC\(_{NN}\) whose pseudocode is reported in Alg.2. Its role is to validate the trajectory provided as input, together with the tuple \((h_0, c_0)\) of the first trajectory state. The outputs of this function are a boolean value (validity) that specifies if the trajectory is valid or not, and the tuple \((h_{end}, c_{end})\), computed for the final state of the validated trajectory. The trajectory to be checked \((\pi)\) is first discretized \((\pi_d)\) using a time step \(\Delta T\) (line 2). Then, all the radii and the inputs profiles \((R_q, R_u, U)\) on this trajectory are predicted by the LSTM network by means of the SensiNN
function (line 3 in Alg. 2). This function takes into account the initial trajectory conditions encoded in the tuple \((h_0, c_0)\), and it returns the above-mentioned radii and inputs, together with the pair \((h_{end}, c_{end})\) by using the LSTM network.

For each state \(q_k\) of the discretized trajectory, the function \(IsValid\) (line 5 in Alg. 2) performs a robust collision checking, testing if there are no collisions despite the state uncertainty, and if the inputs are within some bounds that the system can exert. Since the first operation is known to be computationally expensive, the input bounds are tested first, and, in case they are respected, the collisions are checked between the robot and the environment.

**Algorithm 2 RobustCC\(_{NN}\) \([\pi, (h_0, c_0)]\)**

```plaintext
1: validity ← true;
2: \(\pi_d \leftarrow \text{DiscretizedTraj}(\pi, \Delta T)\);
3: \(\{R_q, R_u, U, (h_{end}, c_{end})\} \leftarrow \text{SensiNN}(\pi_d, (h_0, c_0))\);
4: for each \(q_k\) in \(\pi_d\) do
5: if not \(IsValid(R^h_{du}, U^k, R^k, q_k)\) then
6: validity ← false;
7: break;
8: end if
9: end for
10: return \{validity, \(h_{end}, c_{end}\)\};
```

Note that this robust collision checker can be employed within any tree-based planner. An example on how to include \(\text{RobustCC}\_{NN}\) for the case of a standard RRT planner is given in Alg. 3 with the pseudocode of the \(R\text{-SAMP}\) function, as a particular instance of \(R\text{-SAMP}\) planner. The same notation is used in Sect. V. As shown in lines 7-12 of Alg. 3, a valid output from \(\text{RobustCC}\_{NN}\) simply corresponds to a successful extension from a node, and the final state of the trajectory is inserted in the tree as a new node, storing in it also \(\langle h_{end}, c_{end}\rangle\).

**Algorithm 3 R-SAMP \([qinit, q_{goal}]\)**

```plaintext
1: \(T \leftarrow \{q_{init}, q_{goal}\}\);
2: for \(k = 0\) to \(K\) do
3: \(q_{rand} \leftarrow \text{Sample}()\);
4: \(q_{near} \leftarrow \text{Nearest}(T, q_{rand})\);
5: \(\pi \leftarrow \text{Steer}(q_{near}, q_{rand})\);
6: \((h_0, c_0) \leftarrow q_{near}.\text{getHidden}()\);
7: \{valid, \(h_{end}, c_{end}\)\} \leftarrow \text{RobustCC}_{NN}(\pi, (h_0, c_0))
8: if valid then
9: \(q_{rand}.\text{setHidden}(h_{end}, c_{end})\);
10: \(T.\text{addVertex}(q_{rand})\);
11: \(T.\text{addEdge}(q_{near}, q_{rand})\);
12: end if
13: end for
14: return \(T\);
```

C. Accuracy optimization (A-Optim)

The choice to use a local optimization is justified by the cost function considered in order to optimize the accuracy at desired waypoints. Indeed, the latter is defined as:

\[
c = E[R_q] + \sqrt{\text{var}(R_q)}
\]

which aims at minimizing the average size of the radii and their variance at each desired waypoint. The variance is considered in this cost function so that the minimization of a radius at a given point does not lead to a growth of another radius at another waypoint. This function is then neither additive (i.e., considering two trajectories \((\pi_1, \pi_2)\), the cost of their concatenation \(c(\pi_1) + c(\pi_2)\)) nor monotonic, which makes it impossible to optimize in most global planners. This cost function is then optimized within the A-Optim method while other cost function can be optimized in the R-SAMP (e.g., path length). Given that we do not have the analytic form of the cost function derivatives and that estimating them would be expensive, the accuracy optimization has to be performed by a derivative-free (or black-box) method, e.g., the random shortcut algorithm [26].

IV. APPLICATION TO A 3D-QUADROTOR

In Sect. IV-A, we introduce the scenario on which we tested our framework, using a quadrotor as robotic platform. Then, in Sect. IV-B, we explain how we learned the uncertainty tubes for the quadrotor case.

A. 3D-quadrotor application and planning scenario

The dynamic model of the robot is the same full 3D quadrotor model as described in [3]. The tracking controller considered in this work is the so-called Lee (or geometric) controller [27] where the control inputs are the squared rotor speeds \(u = \sigma^T \omega^2\).

An in-flight ring recovery task is considered for testing our accurate optimization framework. In particular, the quadrotor is tasked to recover some rings by means of a perch mounted along its \(\mathbf{x}\)-axis, as depicted in Fig. 1. The task is performed in the environment shown in Fig. 3, in which the robot must first recover 3 rings in the departure room (red rectangle in the figure), then it must pass through a window of dimension \(1.5m \times 1m\) (blue rectangle) for reaching the second room (green rectangle) and collecting other 4 rings. Finally, it has to return to the departure room, passing again through the small window. In Fig. 3, each ring is represented by a black dot, it has an inner radius of \(0.06m\) and a mass of \(50g\).

For this scenario we consider the uncertain parameters \(\mathbf{p} = [m, g_x, I_x, I_y, I_z]^T\) where \(m\) denotes the mass of the system, \(g_x\) the position of its center of mass along the \(\mathbf{x}\)-axis of the body frame, and \(I_x, I_y, I_z\) are the principal moments of inertia in the body frame. Recovered rings will remain attached to the perch, modifying the mass of the quadrotor along its \(\mathbf{x}\)-axis. This justifies why we considered as uncertain \(g_x\) only and not \(g_y\), due to the fact that the mass and its distribution becomes mostly uncertain along the \(\mathbf{x}\)-axis of the UAV. We chose our nominal parameters as \(\mathbf{p}_\text{n} = [1.5, 0.0, 0.01, 0.01, 0.02]^T\) and their associated uncertainty range \(\delta\mathbf{p} = [10\%, 5cm, 5\%, 5\%]^T\) which represents the percentage variation of the parameters w.r.t. their associated nominal values except for \(g_x\) whose nominal
value is null. The quadrotor trajectories are planned using the Kinosplines steering method of [28], and a RRT* algorithm [20] minimizing the time-length. Robust collision checking is performed by simply considering a bounding sphere of the UAV increased by the predicted state uncertainty radii $R_q$.

B. Learning uncertainty tubes for the 3D-quadrotor

As mentioned in Sect. II-B, our LSTM-based network predicts the uncertainty tubes’ radii and the inputs of the system. In the quadrotor case, the control inputs to be predicted are $U = [U_1, U_2, U_3, U_4]^T$, with $U_i = \omega_i^2$. Regarding the uncertainty tubes, the LSTM-based network predicts $R_q = [R_x, R_y, R_z]^T$ (with $R_k, k \in \{x, y, z\}$ the radius of the state uncertainty tube along the $\{x, y, z\}$-axis) and $R_u = [R_{u1}, R_{u2}, R_{u3}, R_{u4}]^T$ (with $R_{ui}$ the radius of the input uncertainty tube associated with the $i$-th control input of the system).

Then, as shown in eq.(6), the computation of these radii is strongly related to the one of the state/input sensitivity matrices (3–4). Such matrices can be computed numerically by forward integration and thus strongly rely on the Jacobian of the system dynamics. This justifies our choice of using the global velocities and accelerations as well as the yaw angle ($\Psi$) along the trajectory to be the components of $q_i = [\cos(\Psi) \sin(\Psi) q_\delta q_\dot{\delta}]^T$ evaluated at time $i \Delta T$, where $q_\delta = [x y z \Psi]^T$ and $\Delta T$ is the same time step that will be used for the collision checking by the planner.

Finally, the LSTM-based network is trained using the MSE criterion on 10,000 randomly generated trajectories with an execution time of 15s. In order to evaluate the accuracy of the network, we generated an evaluation set composed of 1,000 trajectories whose duration is slightly longer (17s) w.r.t. the ones used in the training set. The performance of the learning is illustrated in Table I that reports the accuracy for each radius according to the L1 error norm.

<table>
<thead>
<tr>
<th>$R_x$</th>
<th>$R_u$</th>
<th>$R_y$</th>
<th>$R_z$</th>
<th>$U_i$</th>
<th>$R_{u1}$</th>
<th>$R_{u2}$</th>
<th>$R_{u3}$</th>
<th>$R_{u4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.96</td>
<td>0.95</td>
<td>0.91</td>
<td>0.99</td>
<td>0.99</td>
<td>0.85</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE I: Accuracy according to L1 norm of the relative error along each component of the predicted vector. $R_{ui}$ is the radius of the uncertainty tube associated with the $i$-th input and $U_i, i \in \{1, 2, 3, 4\}$ represents the $i$-th actuator input.

V. SIMULATION RESULTS

This section first presents results showing the quality of the learning-based computation uncertainty tubes and its high efficiency when used for robust motion planning. Then, we illustrate the ability of the proposed RA-SAMP planning approach to generate high accuracy trajectories for the scenario illustrated by Figure 3.

A. Performance of the learning-based tube computation

Figure 4 compares the learning-based prediction of the tubes radii and inputs wrt. their real values. The curves illustrate the excellent accuracy on the predictions of the
In this paper, we propose a motion planner able to generate trajectories that are intrinsically robust against parametric uncertainties for any robot/controller pair. It is hinged on a LSTM-based learning approach that quickly and accurately estimates the control inputs and the uncertainty tubes of the state and of the inputs. The results confirm the efficiency of the proposed learning method and highlight the impact of integrating it within a motion planner, resulting in a significant reduction of the planning times. Moreover, we showed that our framework is able to locally optimize the planned trajectory in order to minimize the size of the uncertainty tubes of the state at some desired locations, allowing the system to accurately perform a precision task. Future works will focus on considering uncertainties not only at the parametric level, but also on the state estimation, by extending the computation of the tubes for external disturbances. Finally, we aim to experimentally validate the proposed framework on a real robotic platform.

VI. CONCLUSION

<table>
<thead>
<tr>
<th></th>
<th>RRT*</th>
<th>R-SARRT*</th>
<th>RA-SARRT*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure (%)</td>
<td>65.0</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Number of rings picked</td>
<td>±1.6</td>
<td>±3.1</td>
<td>±4.1</td>
</tr>
</tbody>
</table>

An execution failure is characterized by zero ring recovered.
REFERENCES


