A Tuned Approach to Feedback Motion Planning with RRTs under Model Uncertainty

Guilherme J. Maeda, Surya P. N. Singh, Hugh Durrant-Whyte

Abstract—Model uncertainty complicates most kinodynamic motion planning and control approaches due to their reliance on accurate forward prediction. If the model uncertainty is significant, a generated path or control strategy based on forward simulation of this model is potentially invalid and expensive to track (if possible). This paper explores the use of system identification/estimation to tune model parameters. Framed as an extension to rapidly exploring random tree (RRT) methods, it updates the model so that reachable actions added to the tree have more fidelity. This can be viewed as a mixture of a model predictive control (MPC) for local planning with an approximate-model global planner providing sub-goals and thus overcoming the limited lookahead caused by model uncertainty. The benefits of this approach are illustrated for a 3-DOF serial manipulator controlled by computed torque control operating under large external disturbances. In this case, the approach provides operation under intermittent feedback and disturbance observation. Tracking and actuator utilization are also improved over solutions found via conventional methods.

I. INTRODUCTION

For agile planning and efficient control an understanding of the environment is often necessary. Disturbance, noise, and system parameters, such as terrain mobility and motion coefficients, are generally uncertain and vary over time. This is often compensated through a combination of conservative planning and aggressive (high-gain) feedback control with significant control authority at the expense of efficiency. However, even with feedback control, incorrect plans may not always recover due to saturation.

This has motivated significant efforts in the sensing and classification of these parameters. The problem has often been treated empirically, geometrically, or by classification [1], [2]. However, comparatively less research has considered the internal dynamic models used for prediction in general [3]. In kinodynamic planning, such as with the popular Rapidly-exploring Random Tree (RRT) algorithm [4], it is often assumed that the a dynamic model (or a stochastic collection of models [5]) is correct and available, essentially operating as a black box. This gives the RRT a great deal of utility as it facilitates the solution of complex dynamical systems in constrained environments where optimal control is difficult or impossible.

For the case of feedback motion planning an integrated extension to the RRT is presented in which model properties are updated during execution. As illustrated in Fig. 1, environmental disturbance can be estimated resulting in revised open-loop operations that are easier to execute. However, the estimation is essentially an on-line process and needs observations (presumably from robot motion). Thus, a conventional off-line motion plan “expires” and needs recomputation. The model is considered to be subjected to variations that affect the trajectory design, and thus a suitable way to generate the path is by a receding horizon control (RHC) approach. In this way, the uncertainty is not propagated in long horizons; and as the dynamics are being updated, the trajectory is naturally reshaped. The key idea is to grow a local kinodynamic RRT that is biased by sub-goals defined by a prior path (e.g., holonomic RRT search, expert demonstration).

II. RELATED WORK

Uncertain environments are often considered by robust control approaches [6], [7] or by stochastic models [5]. MPC and RHC methods suggest for uncertain systems that parameter estimation is done online (see [8] for overview). This is suitable since every new horizon can be updated by

G. J. Maeda and S. P. N. Singh are with the Australian Centre for Field Robotics, University of Sydney, NSW 2006 Australia. H. Durrant-Whyte is with NICTA, NSW 2015. This work is supported in part by the Australian Research Council (ARC) Centre of Excellence program. Correspondence should be addressed to g.maeda@acfr.usyd.edu.au

©2011 IEEE
a new model and allows for the generation of trajectories that address the dynamics of the current state [9]. Uncertainty is explicitly treated via optimization in the robust MPC framework [10]. Implementation has to be carefully done as non-linear optimization is potentially ill-conditioned, computationally expensive, and, if not done fast enough, unable to respond within the sensing horizon, hence retarding operation.

One difficulty in planning under uncertainty is that a motion planner inevitably propagates prediction error as it expands the path towards the goal. An explicit approach is to vary the decision of the plan based on the likelihood of validity (e.g., obstacle avoidance) [11] or the likelihood of reaching the goal [5], [12]. In the later, a particle filter framework is used to grow RRT extensions as a stochastic process by generating multiple simulations under different conditions, thus obtaining the likelihood of successfully executing each action. This scales unfavorably and can require significant computation and memory in complex problems.

Related to this, the noise/process error can be treated analytically for each plan to be tracked [13]. The initial path generation is decoupled from the uncertainty, and since it does not require simulation, the solutions can be computed readily. In this view, controls are regulated so that the state plus its uncertainty bound remain valid. Its focus is on quantifying the uncertainty, which is distinct from parameter updating, which focuses on plan (re)generation.

This work makes use of a hierarchical planning approach, which is common for autonomous vehicles [14], [15], and can speed computation. Here, an initial dynamically-invariant plan is used to guide the search by biasing the random samples towards its sub-goals. In this initial planning level, there is no consideration of model dynamics and thus it is possible to connect start and goal states with a rough guess as proposed in [16]. However, this generates tracking error and a RRT is grown to execute a local kinodynamic planning during a limited horizon. This structure is particularly suited for long-range planning, but comes with the burden that such approaches are potentially less-informative in cases where motion constraints and dynamics dominate. Furthermore, there is no guarantee that the initial global plan generates feasible sub-goals, and since there is no recomputation of the global planner, this algorithm is not complete. However, because the main concern in this work is uncertainty in dynamics during the second level of planning, obstacle avoidance and path finding in complicated configuration spaces are assumed to be satisfactorily addressed during the initial global planning phase.

III. MODEL UPDATING RRT ALGORITHM

In cases where the optimality conditions can be relaxed, such as when a valid trajectory is sufficient, a RRT approach is computationally efficient, sufficiently reactive (in short horizons), and inherently handles obstacle constraints [4]. However, an incorrect model may result in inefficient or invalid solutions, particularly around performance limits, thus motivating the integration of updating ideas from MPC.

A. The Basic RRT Algorithm

In the RRT algorithm (as introduced in [4]) the BUILD_RRT function initializes the tree and generates random states by sampling in the space. For the case of kinodynamic planning the sampling is done in the state-space and accounts for dynamic constraints, including velocity limits and regions of inevitable collision due to momentum. Actions and their corresponding states are then added to the tree if valid (i.e., collision free, dynamically feasible, etc.). It is important to notice that growing a tree in state-space relies on a good dynamics model.

The EXTEND function receives the random sample $x_{rand}$ and by using a pre-defined sequence of actions, NEW_STATE forward simulates the model in order to predict the future states. These future states are then compared by a metric (often Euclidean distance) to $x_{rand}$, and the closest candidate is selected.

B. Updating the Model with Estimation

In situations similar to those in Fig. 1, unknown reactions or intrinsic parameter variation changes the system dynamics due to model deviations in NEW_STATE, leading to incorrect predictions. This not only generates tracking error, but also decreases planning quality (or even the feasibility). However, those parameters are essentially unknown and must be estimated or sensed as the robot moves. For the same reason, the trees cannot be extended to the goal without factoring for the uncertainty because the dynamics of the robot/environment at the end may be different from the initial values or change on the way. This suggests a receding horizon framework, which allows the model of the planner to be updated constantly while generating the paths as the robot moves.

There are two key benefits to a hierarchical approach. First, under differential constraints a distance metric is not clearly defined and it is known that RRTs do not grow efficiently [17]. However, the efficiency of the search can be increased in certain situations when an initial path is used as a heuristic to guide the search. For example, such a heuristic can be found by a RRT holonomic search over the workspace (see Fig. 2). Second, the discretization size may be used to indicate the range for which the current dynamics are considered valid. These discretization steps do not need to be fixed, but could be made as a function of the amount of parameter variation.

For the proposed algorithm, illustrated in Fig. 3, the input to UPDATED_RRT is an initial global path as the one shown in Fig. 2. Each $x_{subgoal}$ calls the construction of a RRT. Once the tree reaches $x_{subgoal}$ a receding horizon controller (e.g., PID, LQR) tracks the RRT while estimating the dynamics of the model. For the next iteration, the estimated parameters $p_{model}$ are used to update the NEW_STATE function in Algorithm 3 so that the next prediction is based on the newest model estimate. This generates some prediction error, but
Fig. 2. A holonomic search is smoothed and used to seed the RRT expansion for a subsequent nonholonomic search.

Algorithm 1 UPDATED_RRT(x_path)
1: \( P_{model} \leftarrow \text{INITIAL\_MODEL} \)
2: while \( k \neq 0 \) do
3: \( x_{subgoal} \leftarrow \text{SET\_GOAL}(x_{path}) \)
4: \( T \leftarrow \text{BUILD\_RRT}(x_{init}, x_{subgoal}, P_{model}) \)
5: \( P_{model} \leftarrow \text{RHC}(T) \)
6: \( k \leftarrow k - 1 \)
7: end while

Algorithm 2 BUILD_RRT(x_init, x_goal, P_model)
1: \( T.init(x_{init}) \)
2: for \( k = 1 \) to maxIter or subGoalReached do
3: \( x_{rand} \leftarrow \text{RANDOM\_STATE}(N) \)
4: \( x_{reached} \leftarrow \text{EXTEND}(T, x_{rand}, P_{model}) \)
5: if \( x_{reached} \equiv x_{goal} \) then
6: Return goalReached
7: end if
8: end for

Algorithm 3 EXTEND(T, x, P_model)
1: \( x_{near} \leftarrow \text{NEAREST\_NEIGHBOR} \)
2: if NEW\_STATE(x, x_{near}, x_{new}, u_{new}, P_{model}) then
3: \( T.add\_vertex(x_{new}) \)
4: \( T.add\_edge(x_{near}, x_{new}, u_{new}) \)
5: if \( x_{new} = x \) then
6: Return Reached
7: else
8: Return Advanced
9: end if
10: Return Trapped
11: end if

Fig. 3. The Model Updating RRT algorithm (extended from and in the notation of [4]) uses estimation (e.g., Kalman filters) to address the bias caused by the binary addition of actions to the tree by the NEW\_STATE function based on the model (without factoring for control effort, error, etc.).

this is tolerable if the conditions change smoothly or if the planned horizon is short in relation to parameter variation (as shown in Sec. V).

A difficulty that arises with model dynamics variation is that nodes in the previous tree become stale and can not be used for exploration without correction. A random sample that finds its nearest neighbor in an old tree must be ignored (represented by \( x_{rand} \) in Fig. 4). This is naturally solved in the Algorithm 3 because the BUILD\_RRT reinitializes the tree at every new horizon. However, this could decrease efficiency when coverage is important and requires nodes from the past trees (e.g., revisiting an old area needing further investigation). One possible approach is to make the old tree nodes contain the parameters for the model at that position, so that the NEW\_STATE recovers the model parameters if it needs to be revisited.

IV. ILLUSTRATIVE CASE: NONHOLONOMIC ROBOT WITH MODEL UNCERTAINTY

The operation of this approach can be considered for the case of a nonholonomic dynamic robot in a changing environment that is difficult to ascertain in advance. In this case the robot is described by

\[
\begin{align*}
x & = s \cos(\theta) \\
y & = s \sin(\theta) \\
\dot{\theta} & = s/L \tan(u) \\
\dot{s} & = \frac{F}{M} - b/M
\end{align*}
\]

where \( x, y, \theta, s \) represent the position, orientation and speed of the robot, \( L, M, F, u \) are axle distance, mass, force input and steering input. The problem is to generate a sequence of feedforward actions that steer the nonholonomic robot in an environment with obstacles as shown in Fig. 5(a).

Now consider environmental uncertainty, such as if the viscous damping \( b \) is unknown and changing according to the robot position in the \( x \) direction as shown in Fig. 5(b).

A. Standard Kinodynamic Planning & Control

As an initial plan (and for comparison), a RRT is grown to generate commands that drives the robot from the initial position to the goal while avoiding obstacles. Although the condition changes, the tree is grown with a fixed damping value, chiefly, the initial value at the start position.
Decoupled feedback alone is uninformed and less efficient. Three cases of tracking the generated path – from fully open-loop to fully closed-loop – are illustrated in Fig. 6. In Fig. 6(a) the sequences of open-loop actions from the RRT output are applied. The result shows that the commands are actually invalid. As the damping generates different reactions, open-loop commands from the RRT have to be adapted. In Fig. 6(b) a PD feedback controller is used to track the trajectory. The controller attempts to roughly illustrate an intermittent feedback case where actions from the closed-loop control are not available continuously (i.e., to account the time for localization processing). The controller is set so that feedback signal is available for half the cycle (e.g., for 0.25 seconds for a 0.5 second cycle). In Fig. 6(c) the control signal runs continuously. The cumulative distance tracking error in (b) and (c) is 17 m and 5 m, respectively, showing that the execution of the motion planning is highly dependent on the feedback controller.

Although each practical application has its own particularities, similar cases to the one showed in Fig. 6(b) could be given by sensor failures, GPS occlusion, communication delays, or limited bandwidth. In such situations it is desirable to be able to move the robot even without a good feedback controller, however this is only possible if the prediction method accounts for model changes.

B. Model Updating via Estimation

The action of the Model Updating RRT algorithm may be applied to perform parameter tuning. In this case, estimation is accomplished with a linear Kalman filter for the process [8]:

\[ m(t + 1) = m(t) \]  \hspace{1cm} (5)
\[ z(t) = H(t)m(t) + v(t) \]  \hspace{1cm} (6)

where the observation is \( z(t) = s(t + 1) - s(t) - \Delta TF(t)/M \), the parameter to be estimated is \( m = b \), \( H(t) = -\Delta Ts(t)/M \), and \( n(t) \) is a white, zero mean Gaussian random noise. (5) is to indicate that parameter dynamics are assumed static.

Figure 7 shows the simulated result of the previous motion planning problem, where the trajectory generated by the RRT is updated by estimating the current damping. Fig. 8 shows the estimated damping.

In all three cases, the local RRT is guided by the a holonomic search (ref Fig. 2). The three cases are repeated with the same control gains as before (ref. Fig. 6).

The robot is driven with significantly less tracking errors. The open-loop solution shows that the dependency on the feedback controller is minimal and, in practical applications, feedback actions can then be better used to correct other disturbances/errors that are not captured by the estimation.
For the sake of comparison, the velocity profiles and control efforts of cases Fig. 6(c) and Fig. 7(c) are compared in Fig. 9. In the conventional RRT solution, the speed profile increases, since the robot is gaining momentum. The RRT with model updates decreases the speed when the damping increases taking around 3 seconds more to reach the goal when compared to the conventional solution. The control effort of the RRT generated with the fixed model is 33% saturated in relation to the overall execution time, while only 7% in the case the RRT planner is updated.

V. TESTS & RESULTS: MANIPULATION UNDER LARGE DISTURBANCES

The Model Updating RRT algorithm has been implemented and tested for various scenarios. A particularly interesting case is the problem of driving a 3-DOF serial manipulator under high disturbances. Consider, for example, the manipulation problem with the end-effector having to slide a disk amongst obstacles over a surface with variable and unknown stiction levels, as shown in Fig. 10. This friction is reflected at the joints as discontinuous disturbance torques.

Different from the previous case, because the disturbances change abruptly (stiction changes discontinuously according to the opposite sign of velocity), the current dynamics may have a different behavior even in the next lookahead. In such cases the sub-goal discretization that dictates the NEW_STATE horizon must be small.

The NEW_STATE function expands the tree by applying open-loop commands as inputs for driving an unit mass dynamics instead of driving the joints of the manipulator directly. An inner loop uses a computed torque control to map the RRT inputs to the equivalent torques for driving the nonlinear serial manipulator. Under computed torque the RRT expands the tree much sparser than applying the control actions directly on each motor link, since the tree grows free from the coupled nonlinear inertias, centrifugal and gravity forces.

Although the RRT expansion benefits from the dynamic decoupling effect created by the computed torque control,
the sensitivity to modeling errors is considered its main drawback. Parameter estimation [18]-[19], disturbance observation [20], and, more recently, learning methods [21] are used to overcome model errors. This sensitivity causes poor tracking as shown in Fig. 10 where a feedback controller without disturbance observation was used to track the RRT trajectory. The feedback controller was not able to correct the uninformed RRT plan since the feedback controller itself needs parameter adaptation.

In Fig. 11, disturbances effects are minimized by using a disturbance observer (as described in the appendix) applied to the computed torque control. The function NEW_STATE of the RRT local planner is also updated with the same disturbance observations, generating trajectories that reflect the local disturbance (Fig. 12).

VI. CONCLUSIONS AND FUTURE WORK

The presented method supports the view that since the feedback is correlated to the model uncertainty, analyzing this (via system identification) provides means for adapting the model and for better operations. The Model Updating RRT shows that integrating feedback control into the path generation and replanning process provides more efficient operation than feedback alone. As illustrated for a nonholonomic robot and simulated for a manipulator under saturation constraints, this work shows an approach to use estimation of parameters and disturbances in order to improve the quality of an RRT planner and its subsequent control allowing for tracking with less cumulative error and less actuator saturation.

The variation in model dynamics due to intrinsic parameters like inertia and damping, or due to external forces make long term predictions hard to track by feedback alone. Suitable predictions are important to avoid expensive feedback control efforts, or to compensate the lack of feedback when actions have low frequency updates or delays. In computed torque control, the feedback controller itself is sensitive to model variation, thus requiring some form of estimation. It is cheap to use the same estimation to inform the RRT prediction.

Simulated results with the manipulator shows that an RRT local planner for recovery and computed torque method may be a viable option for tasks under high disturbance. While the models in this work afforded the use of a linear estimator, the framework is general and can be adapted. Ongoing work is considering the stability bounds of the estimation with regards to algorithm operation.

A motivating application for manipulation under high disturbance is autonomous excavation. The Autonomous Excavator (shown in Fig. 13) used a fuzzy sliding mode controller to track position set points [22] aiming low level robust control. However, this does not address obstacles (e.g., pipes), crowding, and overcoming significant bogging. Further efforts aim to implement the presented algorithm.
in the experimental excavator platform. The RRT algorithm with model update is expected to add motion planning and trajectory recovery strategies for high disturbance, unknown environment typical of excavation.

VII. ACKNOWLEDGMENTS

This work is supported by the Rio Tinto Centre for Mine Automation and the ARC Centre of Excellence program funded by the Australian Research Council (ARC) and the New South Wales State Government.

VIII. APPENDIX - MANIPULATOR DISTURBANCE ESTIMATION

The general manipulator form is

\[ M(q)\ddot{q} + v(q, \dot{q}) + g(q) = \tau + \tau_d \]  \hspace{1cm} (7)

where \( M \), \( v \) and \( g \) are the mass matrix, centrifugal-Coriolis and gravity vectors, respectively. \( \tau \) is the applied torque from the controller and \( \tau_d \) is the disturbance torque generated from disturbances.

The estimation of disturbance torques can be addressed by considering \( \tau_d \) as a free variable. In this form, the estimation is in fact a disturbance observer. Given \( \tau, q, \dot{q} \) and \( \ddot{q} \), the estimation is done by rearranging (7) to fit the form of (6). In the case \( \tau_d \) is estimated as a free variable:

\[ \tau_d = z = M(q)\ddot{q} + v(q, \dot{q}) + g(q) - \tau \]  \hspace{1cm} (8)

and \( H \) is an identity NxN matrix where \( N \) is the number of estimated parameters.

In practical applications, sources of nonlinear friction and error in parameter values are present in several parts of the machine but are clearly not being considered in (7). Therefore, the estimation of \( \tau_d \) is a lump sum of external reactions and inherent modeling errors. A pure characterization of the external forces as shown in Fig. 12 is only possible in simulation, since the parameters are considered to be perfectly known.

REFERENCES