Sampling Based Model Predictive Control with Application to Autonomous Vehicle Guidance

Damion D. Dunlap
(850) 410-6389
dunlap@eng.fsu.edu
Emmanuel G. Collins Jr.
(850) 410-6373
ecollins@eng.fsu.edu
Charmane V. Caldwell
(850) 410-6389
cvcaldwe@eng.fsu.edu

Center for Intelligent Systems, Control and Robotics (CISCOR)
Florida A&M University - Florida State University College of Engineering (FAMU-FSU COE)
2525 Pottsdamer St. Tallahassee, FL 32310

ABSTRACT
Model Predictive Control (MPC) was originally developed for relatively slow processes in the petroleum and chemical industries and is well known to have difficulties in computing control inputs in real time for processes with fast dynamics. In this paper a novel method called Sampling Based Model Predictive Control (SBMPC) is proposed as a resolution complete MPC algorithm to generate control inputs and system trajectories. The algorithm combines the benefits of sampling based motion planning with MPC while avoiding some of the major pitfalls facing traditional sampling based planning algorithms. The method is based on sampling (i.e., discretizing) the control space at each sample period and using a goal oriented optimization method such as A* in place of standard linear or nonlinear programming or evolutionary algorithms. This formulation of MPC applies easily to systems with nonlinear dynamics and avoids the local minima which can limit the performance of MPC algorithms implemented using nonlinear programming algorithms. Simulation results demonstrate the efficacy of SBMPC in guiding an autonomous ground vehicle from a start posture to a goal posture in regions populated with obstacles.

Keywords
Model Predictive Control, Nonlinear Systems, Sampling, Fast Method

1. INTRODUCTION

Introduced in the late 1970’s, Model Predictive Control (MPC) was developed as a means to systematically incorporate physical and performance constraints into the controller formulation. Due to the fact that traditional MPC generates control updates relatively slowly, MPC has been mostly relegated to the industrial process control industry, where it has seen significant success [8]. Autonomous vehicle guidance can be posed as a kinodynamic motion planning problem, which is defined as solving for an optimal trajectory between a start and an end state while avoiding obstacles and respecting a system model as well as input and state constraints [2]; it is an ideal application for MPC. In order to make MPC applicable to these faster systems, the speed of MPC has to be significantly increased.

There are researchers that have employed MPC for trajectory generation and control of autonomous vehicles. One method [9] uses mixed-integer programming and a process called “cost-to-go” to reduce the prediction horizon and approximate the remainder of the path using a graph search. Each vehicle is modeled as a simple linear point mass with constraints on the speed and turn rate. This method has some relationship to the approach in this paper. However there are key differences revealed. The approach of [11] allows for nonlinear vehicle models and uses MPC coupled with the potential field method to allow the vehicle to maneuver around obstacles and other autonomous vehicles. The approach in this paper does not rely on potential fields and hence is expected to yield trajectories with lower costs.

The integration of sampling based methods into MPC has the potential to further improve the computational speed of MPC. Sampling the set of viable control decisions, effectively reduces the size of the optimization problem while implicitly regarding input constraints through selection of the sampled region.

Initial sampling based planners were extensions of discrete algorithms like best-first search and A*. The randomized potential field method uses random walks in an attempt to guide best-first search out of local minima [1, 4]. The method unfortunately contained quite a few parameters that had to be properly tuned for each new problem, but it did inspire other sampling-based planning methods [5].

Rapidly exploring random trees (RRTs) were developed as an incremental sampling method that yields good results in practice without any parameter tuning. This is accomplished by incrementally constructing a search tree that gradually improves the resolution without explicitly setting any resolution parameters. The tree will eventually cover the free space, \( C_{tree} \), generating a path that regards the constraints. The two major limitations to RRTs as well as all sampling-based motion planners are determining which node is closest to the current tree and which system inputs can be used to make the connection. Determining the closest node involves executing a nearest neighbor (NN) search while the second limitation requires the solution or approximation of a two-point boundary value problem (BVP). There are efficient ways to approximate a NN search but the complexity of the BVP increases significantly with the use of more realistic system models. Various adaptations of the RRT methodology have been created to deal specifically with nonholonomic constraints, dynamic
obstacles as well as to heuristically bias the RRT growth [13, 15, 16].

This paper introduces Sampling Based Model Predictive Control (SBMPC) that is able to use a nonlinear dynamic model, take into account state and input constraints, compute a system trajectory and the associated control inputs for fast systems. The key difference from previous “fast” MPC methods is that the input space is sampled at each sample instant and a goal directed optimization algorithm is used instead of linear or nonlinear programming, or global search methods such as genetic algorithms. SBMPC replaces the potentially costly BVP of traditional sampling methods with integration by sampling in the system input space. This also has the effect of eliminating the need for the NN search due to the fact that the sampled input is applied directly to a current node.

This paper is arranged as follows. Section 2 provides the necessary background to SBMPC. Section 3 presents the SBMPC method. Section 4 presents simulation results. Section 5 discusses the results and makes suggestions regarding future work. Finally Section 6 presents the conclusions.

2. BACKGROUND
The section formulates the kinodynamic motion planning problem and presents the standard principles of MPC, establishing the framework from which SBMPC is constructed.

2.1 The Kinodynamic Motion Planning Problem
A great number of autonomous vehicles utilize an architecture wherein the vehicle guidance task is separated into a motion planning algorithm that generates a trajectory and a feedback controller that tracks the generated trajectory. These systems neglect the fact that most of the motion planning algorithms in current literature do not easily incorporate kinematic or dynamic models [12]. The kinodynamic motion planning problem was formulated to facilitate the development of motion planning algorithms that can more easily reflect the same types of constraints to which the physical systems are subject [2].

Although the kinodynamic motion planning problem is not restricted to a particular type of system, we will assume that the system is the discrete, multi-input multi-output (MIMO), nonlinear system:

\[
x(k + 1) = f(x(k), u(k), w(k))
\]

\[
y^m(k) = g(x(k)) + \xi(k)
\]

where \(x(k)\) is the state, \(u(k)\) is the control input, \(w(k)\) is the unmeasured disturbance, \(y^m(k)\) is the measured output and \(\xi(k)\) is the measurement noise. Given the difficulty of the problem, algorithms that aim for real-time performance must pursue an approximate solution. Accepting an approximate solution necessitates a tradeoff between planning time and optimality, where optimality, although often considered to be minimum time, can also be expressed in other terms such as energy, path length, or closeness to the desired goal state. Given the complexity limitation of practical application, the concepts of sampling were well suited for application in the initial kinodynamic motion planning algorithms.

2.2 Model Predictive Control
Model Predictive Control generally works by solving an optimization problem at every time step \(k\) that determines control inputs for the next \(N\) steps, which is known as the prediction horizon. This optimal control sequence is determined by using the system model to predict the potential system response, which is then evaluated by the cost function \(J\). Most commonly, a quadratic cost function minimizes control effort as well as the error between the predicted and a reference trajectory, \(r\). The prediction and optimization operate together to generate sequences of the control output \(u\) and the resulting system output \(y\).

The various forms of MPC, can generally be classified as either receding horizon [7] or shrinking horizon [6, 14]. Receding Horizon MPC (RH MPC) is primarily used to track a reference trajectory, which usually leads to a set point, while the lesser known Shrinking Horizon MPC (SHMPC) enables the algorithm to generate the trajectory that leads to a given set point or goal region. The latter of these approaches is most applicable to the kinodynamic motion planning problem.

2.2.1 Receding Horizon Model Predictive Control
For RH MPC the optimization problem is typically posed as:

\[
\min J = \sum_{i=1}^{N} \|r(k+i) - y(k+i)\|^2_{Q(i)} + \sum_{i=0}^{M-1} \|u(k+i)\|^2_{S(i)}
\]

subject to the model constraints,

\[
x(k+i) = f(x(k+i-1), u(k+i-1))
\]

\[
y(k+i) = g(x(k+i)) + \beta(k)
\]

and the inequality constraints,

\[
Ax \leq b
\]

\[
C(x) \leq 0
\]

\[
u' \leq u(k+i) \leq u^u
\]

where the prediction and control horizons are \(N\) and \(M\) respectively, \(\beta(k)\) is a bias expression that compares the current predicted output \(y(k)\) to the current measured output \(y^m(k)\). \(C(x)\) represents the nonlinear constraints on the states, and \(Q(i) \geq 0\) and \(S(i) \geq 0\) are the error and control effort weights respectively. The first optimal control input is implemented after which the process is repeated until the goal is achieved. In RH MPC the fixed size prediction window moves at each time step, as illustrated in Fig. 1. The objective is for the predicted output \(y(k+i)\) to reach the reference trajectory \(r(k+i)\), utilizing the proposed inputs \(u(k+i)\) during the prediction horizon \(N\).

2.2.2 Shrinking Horizon Model Predictive Control
SHMPC differs from RH MPC in that it employs a varying size horizon that terminates at the end state. As illustrated in Fig. 2, the prediction horizon “shrinks” as the horizon size decreases by one time interval at each step. In this method, a trajectory is generated that must reach the goal state, \(G\), yielding the optimization problem,
reduce the problem size of MPC by sampling the system input space. The method also replaces the MPC optimization phase with an $A^*$ like algorithm. The SBMPC optimization varies from the typical application of $A^*$ in that the nodes are configurations in the state space and the edges are comprised of the control inputs that link two states. SBMPC retains the computational efficiency and guaranteed convergence properties of $A^*$ [3] while avoiding some of the computational bottlenecks of sampling-based motion planners. The local planning method of sampling-based motion planners is simplified from what is effectively a two-point boundary value problem by sampling in the input space and integrating to determine $q_{new}$. Although most sampling-based planners use proximity to a random or voronoi biased point as their vertex selection method, the proposed method biases the vertex selection with a $A^*$-like heuristic eliminating the need for a potentially costly nearest neighbor search while promoting advancement towards the goal. The resulting algorithm is therefore resolution complete yet its efficiency is dependant on the quality of the heuristic function.

3.1 The SBMPC Algorithm
The main component of the SBMPC algorithm is the optimization, which will be called Sampling-Based Model Predictive Optimization and consists of the following steps:

1. Sample Control Space: Generate a set of samples of the control space that satisfy the input constraints.

2. Generate Neighbor Nodes: Integrate the system model with the control samples to determine the neighbors of the current node.

3. Evaluate Node Costs: Use an $A^*$-like heuristic to evaluate the cost of the generated nodes based on the desired objective (shortest distance, shortest time, or least amount of energy, etc.). The heuristic can be described as optimizing a cost function of the form:

$$\min J = \sum_{i=1}^{N^*-k} \|y(k+i)\|^2_Q + \sum_{i=0}^{M-1} \|u(k+i)\|^2_S$$ (7)

subject to (5) and (6), where $N^*$ is the endpoint, $Q(i) \geq 0$ and $S(i) \geq 0$. The first summation of the cost function minimizes the deviation between the predicted outputs and the endpoint $G$ while the second summation minimizes the control effort. Since the control inputs are chosen that minimize the distance from the goal, the trajectory and necessary control inputs needed to reach the desired endpoint are resolved simultaneously.

Since the longer prediction horizons typically required by SHMPC can be computationally expensive, RHMPC can be made to resemble SHMPC through the use of a low cost prediction beyond the normal prediction horizon as depicted in Fig. 3. The approach was introduced in [10] for application to autonomous vehicle guidance and represents a good paradigm for speeding up SHMPC.

3. SAMPLING-BASED MODEL PREDICTIVE CONTROL
Sampling-Based MPC (SBMPC) combines MPC with the concepts of sampling-based motion planners to effectively
The discretized model is

\[ x(k + 1) = x(k) + \cos(\theta(k)) \cdot u_1(k) \cdot T_s \]
\[ y(k + 1) = y(k) + \sin(\theta(k)) \cdot u_1(k) \cdot T_s \]
\[ \theta(k + 1) = \theta(k) + \frac{\tan(\psi(k))}{l} \cdot u_1(k) \cdot T_s \]
\[ \psi(k + 1) = \psi(k) + u_2(k) \cdot T_s \]

where \( u_1 \) and \( u_2 \) are the forward velocity and steering rate respectively, \( l \) is the longitudinal wheel separation, \( \psi \) and \( \theta \) are the steering and orientation angles respectively, and \( T_s \) is the time step. The non-holonomic constraint can be expressed as:

\[ x(k + 1) \sin(\theta(k)) - y(k + 1) \cos(\theta(k)) = 0 \]

In order to illustrate the properties of SBMPC, the algorithm was applied to randomly generated scenarios with 50 obstacles of various sizes. The velocity and steering rate were constrained as indicated in Table 1 and all the scenarios begin at the origin, \((0,0)\) and end at \((10,10)\). The thin segments represent all the trajectories that were evaluated while the thicker line represents the calculated trajectory. The first and second scenarios in Figs. 4 and 5 are representative of typical results where the resulting path consists of 12-19 steps with computation times below 0.8s. The smoothness of the trajectory indicates how the steering rate constraint limits the maneuverability of the vehicle, forcing it to make wider turns than would occur without this constraint.

The third scenario shown in Fig. 6 demonstrates the ability of SBMPC to negotiate potential local minimum as well as navigate narrow passages. The algorithm continues to generate samples until a solution path is found and although this does result in longer computation times, the algorithm has never failed to generate a valid solution.

The final scenario, shown in Fig. 7, highlights two of the main problems with the current implementation of SBMPC, the most obvious being the necessity of increasing the sampling near the goal region as indicated by the density of potential paths shown in that area. The second is the backtracking that occurs near the middle of the path as well as near the goal region. This occurs due to the absence of what is referred to in the \( A^* \) algorithm as a back pointer. The addition of a back pointer would enable the algorithm to identify when multiple trajectories lead to same state. The path to that state can then be redefined to be the lowest cost of the paths. Future implementations of SBMPC will integrate the back pointer which should make the algorithm more efficient while improving the final solution.

<table>
<thead>
<tr>
<th>Table 1: Simulation Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sampling Time</strong></td>
</tr>
<tr>
<td><strong>Subsampling Time</strong></td>
</tr>
<tr>
<td><strong>No. of Samples per expansion</strong></td>
</tr>
<tr>
<td><strong>Velocity:</strong> ( u_1 )(m/s)</td>
</tr>
<tr>
<td><strong>Steering Rate:</strong> ( u_2 )(rad/s)</td>
</tr>
</tbody>
</table>

In order to illustrate the properties of SBMPC, the algorithm was applied to randomly generated scenarios with 50 obstacles of various sizes. The velocity and steering rate were constrained as indicated in Table 1 and all the scenarios begin at the origin, \((0,0)\) and end at \((10,10)\). The thin segments represent all the trajectories that were evaluated while the thicker line represents the calculated trajectory. The first and second scenarios in Figs. 4 and 5 are representative of typical results where the resulting path consists of 12-19 steps with computation times below 0.8s. The smoothness of the trajectory indicates how the steering rate constraint limits the maneuverability of the vehicle, forcing it to make wider turns than would occur without this constraint.
on computational performance and solution quality. The current implementations of the MPC variants are coded in Matlab while SBMPC is implemented in C++, which would not offer a fair comparison. Preliminary comparison results have shown SBMPC’s ability to successfully negotiate scenarios where the Newton based method gets stuck in a local minimum. The simplicity of SBMPC has also made it more capable of obtaining quality solutions for various scenarios than the other methods.

SBMPC effectively makes the trade off between computation time and solution optimality but if there are not enough samples generated near the goal region, SBMPC can be inefficient in reaching the goal state. For example, the goal region shown in Fig. 8 may require several more iterations to reach the goal region when the goal could have been achieved by the current step if more appropriate samples had been generated. The illustration was generated by evaluating all possible control sequences of length 2 from a fixed set of inputs and also demonstrates how this strategy could fail to reach the goal.

5. DISCUSSION
The current SBMPC software was designed primarily for usability and has not yet been optimized for speed. The optimized code is expected to yield much faster computational times; in fact, given that the main computational difference between the SBMPC algorithm and the A* and D* algorithms involves the integration of the model, the performance of SBMPC is not expected to be substantially slower than standard uses of A* and D*. The two main areas of improvement in the current software involve integration of some type of spatial hashing of the state or output space and substantial improvement of the sorting of the Open List. To illustrate the need for spatial hashing, consider the looping behavior shown before the path enters the narrow passage of Figure 6. Spatial hashing would allow the algorithm to identify when a particular state is reached by two separate trajectories and enable the algorithm to choose the lower cost path, thereby eliminating the unnecessary path segments. Currently, most of the computational time is used to sort the Open List, which can be significantly improved through the use of more efficient data structures that reduce the amount of data that is actually moved around during the sorting process. In addition, fast replanning has not yet been implemented in the current software.
The SBMPC algorithm is resolution complete (see Section 3). Practically, this means that if the algorithm is allowed to run, which may involve adding more samples at some time steps, it will find a solution if it exists. The solution will be globally optimal for the sampling assumed. This is true whether the model is linear or nonlinear, which is a very powerful feature of the algorithm.

In addition, the algorithm is extremely easy to tune. The only significant tuning parameter is the default number of samples at each sample period. In contrast, standard MPC requires tuning of the prediction and control horizons and for nonlinear systems can be sensitive to the choice of the initial input trajectory; in the experience of the authors it is more difficult to tune in order to achieve good performance. Both methods also require that the cost function weights be chosen. The relative effect of the weights on the two approaches is currently unclear. A comprehensive comparison of the tuning of the two approaches is part of future research.

6. CONCLUSION
Sampling Based Model Predictive Control has been shown to effectively guide autonomous vehicles incorporating the vehicle model to determine the optimum trajectory. SBMPC exploits sampling-based concepts along with the A* optimization algorithm to achieve the goal of being able to quickly determine control updates while avoiding local minima and regarding all constraints. SBMPC addresses the kinodynamic motion planning problems of generating a trajectory, but can also be used to follow a trajectory as in Receding Horizon MPC by selecting coincidence points along the trajectory and treating each as an endpoint.

7. DISCLAIMER
The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government. References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

8. REFERENCES