Robust Sampling-Based Trajectory Tracking for Autonomous Vehicles

Aneesh Sharma
Email: as10ac@my.fsu.edu

Camilo Ordonez
Email: co04d@my.fsu.edu

Emmanuel G. Collins, Jr
Email: ecollins@eng.fsu.edu

Center for Intelligent Systems, Controls and Robotics (CISCOR)
Department of Mechanical Engineering
Florida A&M University-Florida State University
Tallahassee, Fl 32310

***Abstract*—In real world motion planning tasks, autonomous vehicles can easily deviate away from their planned trajectories due to external disturbances, uncertain wheel/leg-terrain interaction, and other errors in the model used for planning. A possible solution to this problem consists in the continuous usage of replanning strategies. However, replanning is in general computationally intensive and its use should be minimized when possible. In this paper, a new methodology for robust trajectory tracking is proposed. The method generates, via sampling, correcting control inputs to drive the vehicle back to the desired trajectory. Due to the use of sampling, the methodology easily incorporates nonlinear planning models and integrates seamlessly with sampling-based motion planners. The paper presents simulation and preliminary experimental results showing the efficacy of the proposed approach and thus its potential application to motion planning tasks with real-time constraints.**

I. Introduction

One of the core problems in autonomous vehicle mobility is the integration of motion planning and control [1]. Motion planning refers to finding a collision free trajectory in space for a given task, while control aims to ensure that the vehicle follows the planned trajectory. In a simplistic open loop strategy, a vehicle model is used to compute a trajectory and the corresponding control sequence that will result in the vehicle moving from the start state to the goal state. The vehicle is then commanded with the control sequence and if the model has high fidelity, the vehicle will move in the desired trajectory.

Of course, in real world applications pose errors appear due to model errors and the vehicle will not generally move in the desired trajectory when commanded with the original control sequence. Excessive error can lead to unsafe operation, or even loss of control in cases of limited actuation [2]. Even small intermediate pose errors, if not corrected promptly, can result in a large error in the vehicle's final pose. There are various factors which can cause deviation from the planned trajectory such as slippage, uncertain wheel/leg-terrain interaction, actuator limitations, and other errors or omissions in the planning model. As a result, the reference trajectory becomes invalid because the planned control inputs are no longer capable of taking the vehicle to the desired goal.

Conventional motion planning approaches [3], [4], [5], [6] require two disconnected steps to solve this problem: 1) plan a vehicle trajectory, and 2) develop trajectory following controllers. The feedback laws of element 2 are in general complicated, particularly when the system is governed by nonlinear models. Another possible solution to this problem is to replan whenever the vehicle deviates from its trajectory [7], [8], [9]. One particular advantage of using the latter strategy is that it results in close to optimal trajectories from the vehicle's current state to the goal state. While the replanning approaches of [7], [8], [9] address environmental uncertainty due to obstacles, these approaches do not address the issue of uncertainty in the vehicle's position. If a new trajectory is replanned from the current position to the goal position each time the vehicle deviates from its reference trajectory, then the motion planning task becomes computationally very intensive for many unstructured real-world scenarios. A preferred solution would then be to merge onto the existing trajectory in a close to optimal manner.

In this paper we describe a sampling-based algorithm that robustly tracks planed trajectories while optimizing conventional measures of performance such as execution time, length of path, or energy. The algorithm's key features are its low computational requirements, its ability to work with nonlinear models, and its seamless integration with sampling-based motion planners. The remainder of the paper is structured as follow. Section II provides background for the employed motion planner. Section III details the main steps of the proposed algorithm. Section IV presents simulation results. Section V provides experimental results. Finally, Section VI provides conclusions and future work.

II. Background

This section briefly discusses the sampling-based motion planner utilized in this research to generate the baseline trajectory for the autonomous vehicle to follow.

Sampling Based Model Predictive Optimization (SBMPO) [10],[11] may be classified as a randomized A* algorithm that samples exclusively in the input space of the discrete-time model that it integrates; this paradigm is compatible with the input-centric viewpoint of Model Predictive Control [12]. The core optimization is that of the LPA* (Lifelong Planning A*) graph search algorithm and hence, under the conditions that an implicit grid is used for the entire state vector of SBMPOs integration model, SBMPO shares the powerful completeness properties of LPA* [8]. This result is akin to MPC stability results based on enforcement of a terminal

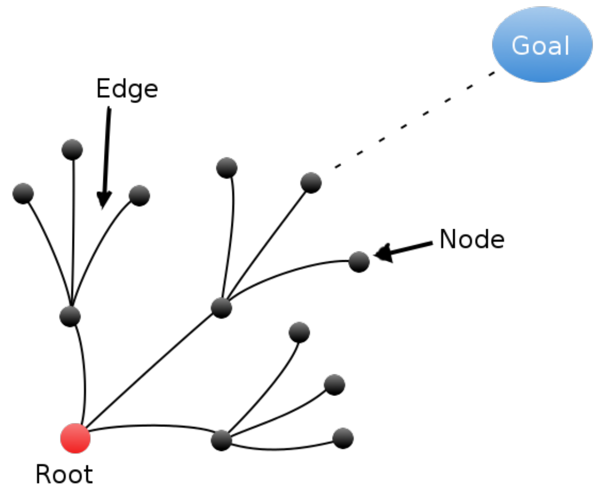

Fig. 1. Illustration of a tree resulting from the SBMPO sampling process.

constraint [12]. To achieve efficient computations, SBMPO utilizes enhanced kinematic models as the integration model and extracts constraints from the vehicle's dynamic model [10], [13], [14]. SBMPO, as do all A* algorithms, requires an optimistic heuristic, which despite its name is actually a rigorous lower bound on the cost from the current state to the goal state. If properly chosen, SBMPO is computationally fast.

The following are the main steps of SBMPO:

1) *Select a node with highest priority in the queue:* The nodes are collected in an Open List, which ranks the potential expansion by their priority or low cost associated with the node. The Open List is implemented as a heap so that the highest priority node that has not been expanded is on top. If the selected node is the goal, SBMPO terminates, otherwise go to step 2. Note that the node representing the start will have the highest initial priority.
2) *Sample the input space:* Generate a sample of the input to the system that satisfies the input constraints. The input sample and current state (i.e., the state of the selected node) are passed to the system model, and the system model is integrated to determine the next state of the system. If the next state satisfies all constraints, then continue to Step 3, else repeat Step 2.
3) *Add a new node to the graph:* Use an implicit grid ([15]) to check if the graph already contains a node close to the new state of the system. If such a node exists, only add an edge from the current node (i.e., the selected node) to the node whose state is similar to the new state. Otherwise, add a node whose state is the next state.
4) *Evaluate the new node cost:* Use an A* heuristic to evaluate the cost of the generated vertices based on the desired objective (which is least amount of energy). Add a new node to the priority queue based on the minimum cost.
5) *Repeat $2-4$ for n successors:* Repeat steps $2-4$ for n successors, where n, the *branchout factor*, is defined by the user.
6) *Repeat $1-5$ until one of the stopping criteria is true:* Steps $1-5$ are repeated until the goal is reached or the maximum number of allowable iterations is achieved.

III. Robust Trajectory Tracking

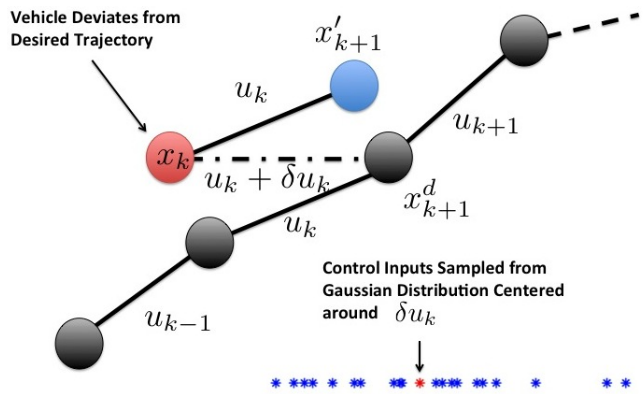

Fig. 2. The proposed approach to trajectory merging. Linearization is used to compute a corrective input δu_k. Samples about $u_k + \delta u_k$ are then input to the nonlinear model. A state, denoted by x, is represented by a circle and a control input, denoted by u, is represented as an edge connecting two states.

Let us assume that as as depicted in Fig. 2, a vehicle trajectory has been planned, but during execution the vehicle realizes that it has diverged from the desired trajectory. In addition, we can assume that the vehicle motion is governed by a general nonlinear model

$$x_{k+1} = f(x_k, u_k), \tag{1}$$

where x_k represents the system state and u_k the control input. The sampling-based planning algorithm described in Section II generates vehicle trajectories by sampling the control inputs u_k. Here, along the same lines, it is proposed that correcting control inputs are generated to drive the vehicle back to the desired trajectory.

The proposed approach relies on linearization of the system around the current state to generate an initial (rough) estimate of a corrective perturbation δu_k of the planned control input u_k. Referring to Fig. 2, if the control input corresponding to the nearest neighbor to the actual state of the vehicle were applied, the vehicle state would change as

$$x'_{k+1} = f(x_k, u_k) \approx Ax_k + Bu_k, \tag{2}$$

where A and B are respectively the corresponding system and input matrices of the system obtained through linearization of (1). Alternatively, if a control input $u_k + \delta u_k$ were applied, the system would evolve as

$$x^d_{k+1} \approx Ax_k + B(u_k + \delta u_k). \tag{3}$$

Therefore, if we define $\delta x = x^d_{k+1} - x'_{k+1}$, the required control perturbation δu_k to drive the vehicle back to the trajectory can be chosen as the least squares solution (δu_k) to

$$B\delta u_k = \delta_x. \tag{4}$$

SBMPO then samples around $u_k + \delta u_k$ and inputs these samples to the full nonlinear model, making the approach a nonlinear one, despite the initial linearization.

To compensate for a pose deviation, the proposed methodology first finds a point of entry ($node_{poe}$) on the reference trajectory. This point, as the name suggests, is a node on the reference trajectory where the vehicle attempts to merge. Ideally this node lies somewhere between vehicle's expected position ($node_e$) and the goal ($node_{goal}$). If it is too close to $node_e$, then the vehicle will have to make very sharp turns, which is highly inefficient because sharp turns introduce more slip-related errors and can consume excessive energy [16], [17], [18]. Secondly, if the point is too close to $node_{goal}$, then there is a high probability that the vehicle may never merge on to the reference trajectory. This can result in significant increase in execution time if the goal is far away from vehicle's current position.

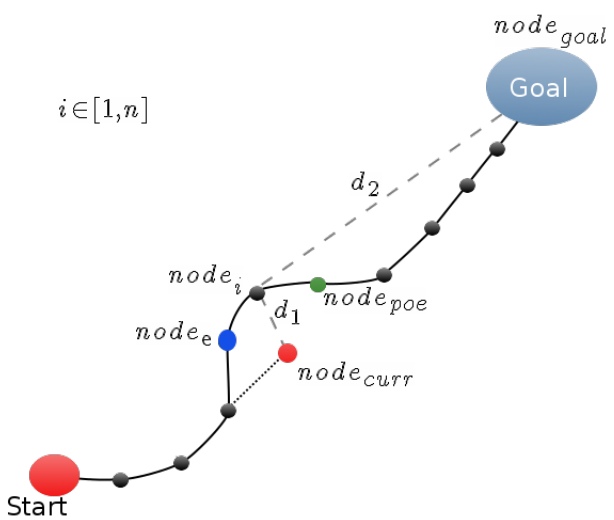

Fig. 3. Sample curvilinear trajectory. The point $node_e$ represents the node where vehicle was expected to be and $node_{curr}$ is the vehicle's current node

Referring to Fig. 3, a distance (d_i) between the vehicle's current position $(node_{curr})$ and the i^{th} node in the reference trajectory $(node_i)$ is defined as the weighted sum of d_1 and d_2,

$$d_i = w * d_1 + (1 - w) * d_2, \tag{5}$$

where w is the weight, and d_1, d_2 are the Euclidean distances between $(node_{curr}, node_i)$, and $(node_i, node_{goal})$ respectively. As the weight w varies from 0 to 1, $node_{poe}$ moves from $node_{goal}$ to $node_e$ with $node_{poe} = node_{goal}$ at $w = 0$, and $node_{poe} = node_e$ at $w = 1$. The ideal point of entry $(node_{poe})$ is selected as the one with the minimum distance $d = minD$, where $D = \{d_i\}_{i=1}^{n}$ and n is the number of nodes in the reference trajectory.

In order to merge onto the reference trajectory at $node_{poe}$, (2)-(4) are applied to compute an estimate of a corrective perturbation δu_{curr} that drives the system towards the desired node of entry. To compensate for the errors caused by the linearization process, the proposed approach utilizes Gaussian sampling around $u_{curr} + \delta u_{curr}$. Sampling gives a more accurate value to the new control input, and is highly efficient in scenarios where one or more obstacles are present around $node_{curr}$.

The planner creates a dynamic directed graph G, which is a set of all nodes, indicating different sample values of $u_{curr} + \delta u_{curr}$, and edges currently in the graph. The cost of traversing from node n' to node $n \in Successor(n')$[1] is denoted by $c(n', n)$, where $0 < c(n', n) < \infty$.
Referring to Fig. 4, the cost between (x_k, x_k') is given by

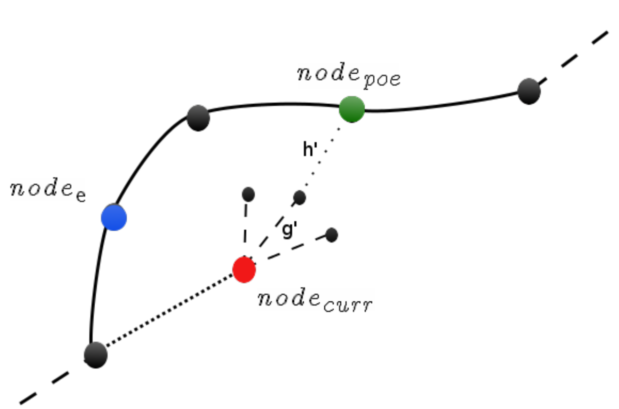

Fig. 4. Illustration of sampling around $u_k + \delta u_k$. The point $node_e$ represents the node where the vehicle was expected to go, $node_{curr}$ is the vehicle's current node, and $node_{poe}$ is the desired node of entry.

$g'(k)$, and the cost from (x_{k+1}', x_{k+1}^d) is denoted by $h'(k)$. The nodes are selected based on their priority, and it is determined by a two component key vector,

$$key(v) = \left(\frac{k_1(v)}{k_2(v)}\right) = \left(\frac{g'(v) + h'(v)}{g'(v)}\right),$$

where $0 \le v < (number\ of\ nodes)$, and the keys are ordered lexicographically with the smaller key values having a higher priority.

In the proposed methodology, the algorithm looks ahead and verifies if it is possible to reach the reference trajectory in less than N steps. If *yes*, then it proceeds with merging strategy (Note that for the merging computations a maximum allocated time T_{ME} of the order of microseconds is preassigned.) Otherwise, a new trajectory needs to be replanned using an anytime version of SBMPO. The incorporation of the anytime version of the planner is part of ongoing work. However, for completeness Fig. 5, provides a schematic of the overall approach.

IV. Simulation Results

In the following simulations a unicycle kinematic model is assumed, and is given in discrete time by

$$\begin{aligned} x_{k+1} &= x_k + v_k \cos(\theta_{k+1})T, \\ y_{k+1} &= y_k + v_k \sin(\theta_{k+1})T, \\ \theta_{k+1} &= \theta_k + \omega_k T, \end{aligned} \tag{6}$$

where (x, y) is the position of the vehicle in the global coordinate frame of Fig. 6, v is the linear velocity, ω is the

[1] $Successor(n')$ denotes the set of successors (children) of vertex $n' \epsilon G$.

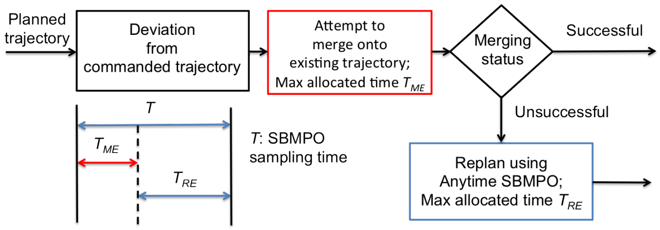

Fig. 5. The proposed approach to robust trajectory tracking with a maximum allocated time T_{ME} for merging, of the order of microseconds. If the merging fails, the anytime version of SBMPO will replan a new trajectory and optimize it for the remaining time T_{RE} (fraction of a second).

angular velocity and θ is the vehicle's orientation with respect to the X axis. The control inputs of the vehicle are (ω, v), and the states are sampled with period T. In addition, the selected cost function is distance; however the approach can easily use other cost functions such as energy [17] or time [14].

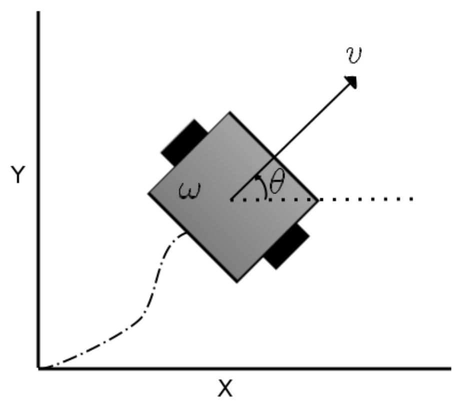

Fig. 6. Coordinate frame used in the simulations. The vehicle 's linear and angular velocities are v and ω respectively

In the simulations, SBMPO is used to generate the reference trajectory. The control inputs from the trajectory are then given to the vehicle in a sequential manner, which if executed properly will take the vehicle to the desired goal state. Figure 7 shows a situation where the vehicle deviates (an external deviation error at $(-0.5, 1.5)$ was introduced) from the desired trajectory and the sampling-based merging approach is not employed. Notice the difference between the reference and not-corrected trajectories. If the planner replans every time the vehicle deviates, then a trajectory like the one shown in Fig. 8 is obtained.

However, it is important to recall that replanning is expensive, which is one of the main motivations for the proposed approach, which intelligently makes a choice between merging and replanning. Figure 9 shows the trajectory generated by the algorithm proposed in Section III (computation time = $8ms$), which quickly estimates the new control inputs required to merge onto the reference trajectory and replans only when it is necessary, thereby reducing the computation time required to generate a corrected trajectory. The computation time of the trajectory merging approach is $0.09ms$ while time taken by replanning approach is $5ms$. This clearly shows that replanning can be a computationally expensive choice in some cases.

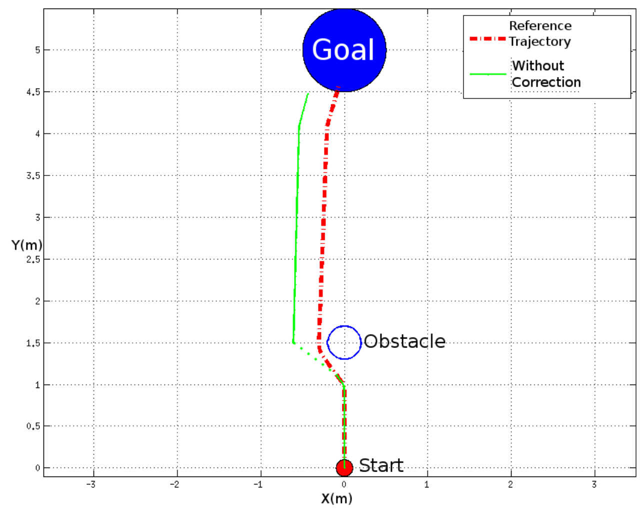

Fig. 7. Simulation result for simplistic open loop control. The dotted green line represents the continued deviation from the dashed-dot red reference trajectory after a perturbation occurs from the desired trajectory

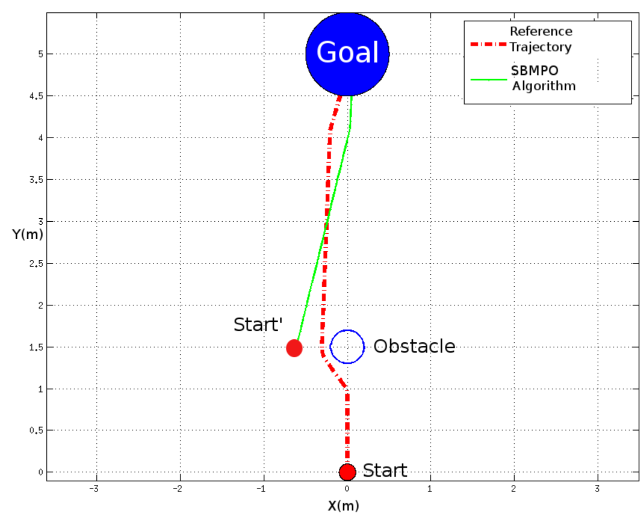

Fig. 8. Simulation result for complete replanning from the perturbed position using SBMPO. $Start'$ is the perturbed position from where a new trajectory is planned to the Goal.

V. Experimental Results

The FAMU-FSU Bot shown in Fig.10 is a skid-steered vehicle, which employs 2 mechanically coupled Pittman GM 9236 brushed DC motors per side. Each pair of motors is controlled using a current control approach by a Maxon motor controller (4-Q-dc). The motor controllers are configured to provide a maximum current of 5A, which corresponds to a maximum torque of about 4.35Nm. The range of angular velocity has been constrained to $-0.33rad/s$ to $0.33rad/s$, with a maximum linear velocity of $0.2m/s$. The motion of the vehicle is tracked using 10 camera Vicon systems.

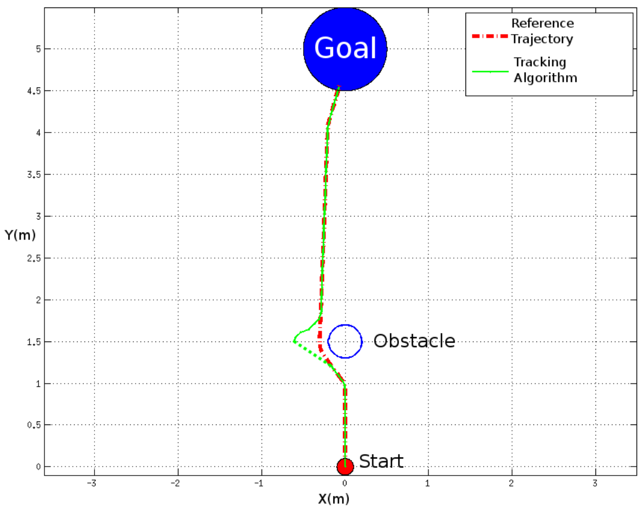

Fig. 9. Simulation result for the proposed robust trajectory tracking algorithm. The dotted green line, representing deviation from the dashed-dot red reference trajectory, converges quickly back to the original trajectory.

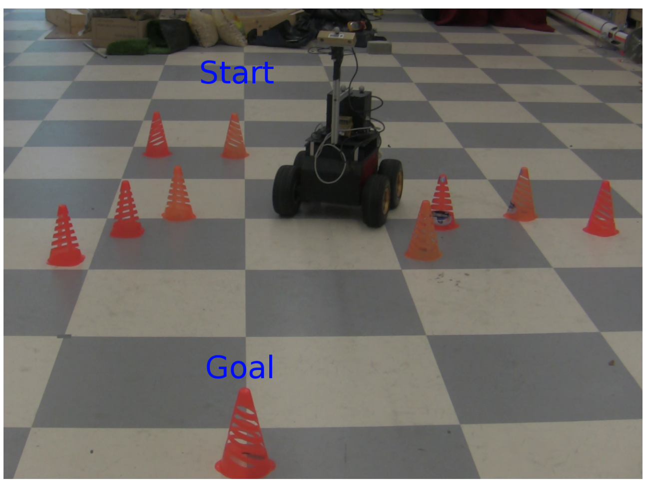

Fig. 10. Experimental setup. The task was to command the FAMU-FSU Bot from the Start position to the Goal position without hitting any of the obstacles.

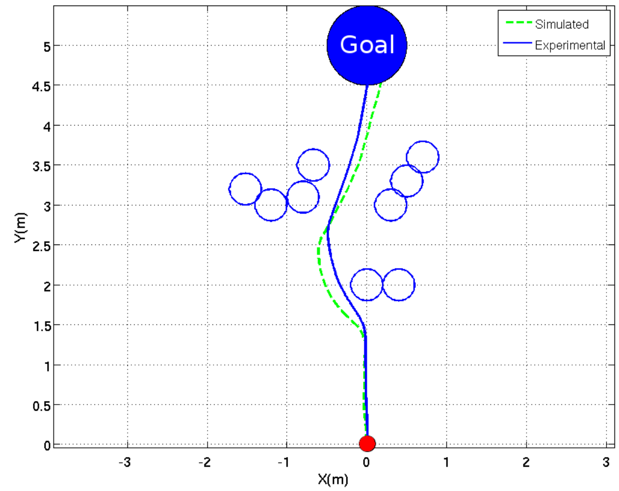

Fig. 11. Original experimental trajectory. The dashed green line represents the simulated trajectory and the solid blue line represents the trajectory executed by the vehicle.

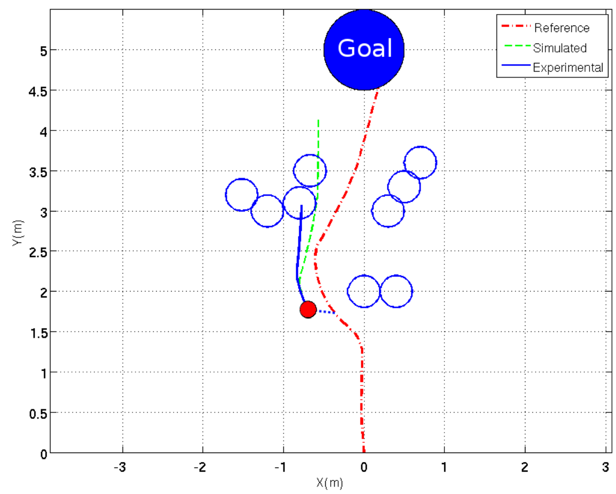

Fig. 12. Trajectory with no correction. The dashed-dot red line represents the reference trajectory, the dotted blue line the deviation from the reference trajectory, the solid blue line the trajectory executed by the vehicle, and the dashed green line the simulated trajectory.

In the experiment, SBMPO used the kinematic model (6) to generate the reference trajectory and the corresponding control sequence. The vehicle's position was tracked using a Vicon camera system. Figure 11 shows an experimental run with the original control sequence. The computation time required to generate the reference trajectory was $20ms$. The difference between the experimental trajectory and simulated trajectory was caused by vehicle slippage. Figure 12 shows a situation where an error of the order of $0.6m$ was introduced manually to the vehicle's position at $(-0.3, 1.7)m$ and the sampling-based merging approach was not applied, resulting in the vehicle colliding with an obstacle. (The vehicle was stopped by a remote off button once it hit an obstacle.) When the sampling-based merging approach was used in the same scenario, the results shown in Figure 13 were obtained. The computation time of the trajectory merging approach was only $0.1ms$.

VI. Conclusion and Future Work

This paper has presented a robust trajectory tracking algorithm for autonomous vehicles. The approach is based on sampling and therefore can efficiently work with nonlinear models without the need to resort to complex feedback laws that are decoupled from the motion planner. In contrast, the proposed approach seamlessly integrates with sampling-based motion planners.

Simulation and preliminary experimental results show the efficiency and efficacy of the approach to drive the vehicle back to a reference trajectory in the presence of external disturbances. In particular the computation times of the trajectory merging approach can be well over an order of magnitude lower than the computation times demanded by complete replanning.

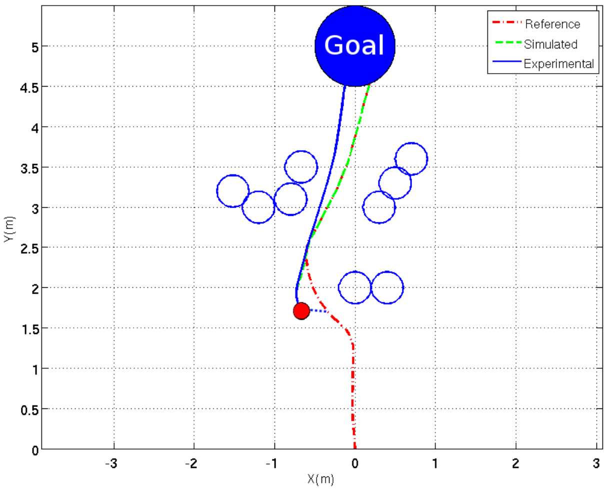

Fig. 13. Experimental result of SBMPO trajectory tracking algorithm; Dotted blue line represents deviation from the reference trajectory

Ongoing and future work involves the development and incorporation of an anytime version of SBMPO, which will enable the replanning of trajectories on demand when the pose errors are very large and the merging approach becomes infeasible. In addition, a detailed experimental validation of the approach will be conducted on outdoor surfaces. Special emphasis will be placed on autonomous ground vehicles moving on very slippery surfaces. Additionally, extensive simulations will be performed for autonomous rendezvous of a spacecraft moving in a cluttered environment [13], which involves a planning problem that relies on direct integration of an uncertain dynamic model.

ACKNOWLEDGEMENT

This work was supported by the Federal Aviation Administration under Grant CA 10-C-CST-FSU and by the collaborative participation in the Robotics Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD 19-01-2-0012. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation thereon.

REFERENCES

[1] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. The MIT Press, 2011.

[2] C. Ordonez, N. Gupta, W. Yu, O. Chuy, and E. Collins, "Modeling of skid-steered wheeled robotic vehicles on sloped terrains," in *ASME Dynamics Systems and Control Conference*, Ft Lauderdale, Fl, October 2012.

[3] S. Blai, "A novel trajectory-tracking control law for wheeled mobile robots," *Robotics and Autonomous Sytems*, vol. 59, no. 2, pp. 1001–1007, 2011.

[4] D. Chwa, "Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates," *IEEE Transactions on Control System Technology*, vol. 12, no. 4, pp. 637–644, 2004.

[5] B. S. Park, S. J. Yoo, J. B. Park, and Y. H. Choi, "A simple adaptive control approach for trajectory tracking of electrically driven nonholonomic mobile robots," *IEEE Transactions on Control System Technology*, vol. 18, no. 5, pp. 1199–1206, 2010.

[6] I. Zohar, A. Ailon, and R. Rabinovici, "Mobile robot characterized by dynamic and kinematic equations and actuator dynamics: Trajectory tracking and related application," *Robotics and Autonomous Systems*, vol. 59, no. 6, pp. 343–353, 2011.

[7] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm." in *ICAPS*, 2005, pp. 262–271.

[8] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 1, pp. 93–146, 2004.

[9] D. Ferguson and A. Stentz, "Anytime RRTS," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 5369–5375.

[10] O. Chuy, E. Collins, D. Dunlap, and A. Sharma, "Sampling-based direct trajectory generation using the minimum time cost function," in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics, J. P. Desai, G. Dudek, O. Khatib, and V. Kumar, Eds. Springer International Publishing, 2013, vol. 88, pp. 651–666.

[11] D. Dunlap, C. Caldwell, and E. Collins, "Nonlinear model predictive control using sampling and goal-directed optimization," in *Proceedings of the Multi-Conference on Systems and Control*, Yokohama, Japan, September 8-9 2010.

[12] J. Maciejowsky, *Predictive Control with Constraints*. Prentice Hall, 2002.

[13] G. D. Francis, E. G. Collins Jr, O. Chuy, and A. Sharma, "Sampling-based trajectory generation for autonomous spacecraft rendezvous and docking," 2013.

[14] C. Ordonez, N. Gupta, O. Chuy, and E. Collins, "Momentum based traversal of mobility challenges for autonomous ground vehicles," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 6-10 2013.

[15] C. Ericson, *Real–Time Collision Detection*, D. H. Elberly, Ed. Elsevier, 2005.

[16] W. Yu, E. Collins, and O. Chuy, *Dynamic Modeling and Power Modeling of Robotic Skid-Steered Wheeled Vehicles, Mobile Robots - Current Trends, Zoran Gacovski (Ed.)*. InTech, 2011. [Online]. Available: http://www.intechopen.com/books/mobile-robots-current-trends/dynamic-modeling-and-power-modeling-of-robotic-skid-steered-wheeled-vehicles

[17] C. Ordonez, N. Gupta, E. G. Collins, J. Clark, and A. M. Johnson, "Power modeling of the XRL hexapedal robot and its applications to energy efficient motion planning," in *International Conference on Climbing and Walking Robots (CLAWAR)*, July 2012.

[18] O. Chuy, E. Collins, W. Yu, and C. Ordonez, "Power modeling of a skid steered, wheeled robotic ground vehicle," in *Proceedings of the IEEE Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 4118–4123.