ACCOUNTING FOR PARAMETRIC MODEL UNCERTAINTY IN COLLISION AVOIDANCE FOR UNMANNED VEHICLES USING SPARSE GRID INTERPOLATION

Sarah L. Noble
Joel M. Esposito
Jason Case
Systems Engineering
United States Naval Academy
Annapolis, MD 21401 USA
{noble,esposito}@usna.edu

ABSTRACT

In this paper we present an enhancement of model-based trajectory selection algorithms – a popular class of collision avoidance techniques for autonomous ground vehicles. Rather than dilate a set of individual candidate trajectories in an ad hoc way to account for uncertainty, we generate a set of trajectory clouds – sets of states that represent possible future poses over a product of intervals representing uncertainty in the model parameters, initial conditions and actuator commands. The clouds are generated using the sparse-grid interpolation method which is both error-controlled and computationally efficient. The approach is implemented on a differential drive vehicle.

INTRODUCTION

Many autonomous ground vehicles (AGVs) employ the multi-layer motion planning framework illustrated in Fig. 1. The high-level planner (a.k.a. mission or route planner) generates a series of way points, or sub-goals, using an *a priori* map reflecting the location of fixed obstacles, the presence and directionality of roadways and other mission constraints. A collision avoidance algorithm (a.k.a. trajectory generator or low-level planner), then generates admissible steering inputs to progress toward this way point, while avoiding collisions or undrivable terrain. At the lowest level, these steering commands are usually then implemented by a PID-based steering controller. Nearly every entry in the DARPA Grand [1] and Urban Challenges [2] and European Land Robotics Trials employed this multi-level planning paradigm.

In this paper we focus on a particular class of collision avoidance algorithms referred to here as *model-based trajectory selection algorithms*. Such approaches rely on a kinematic or dynamic model of the vehicle to evaluate a set of candidate steering inputs, based on the predicted trajectory of the vehicle.

One major limitation of model-based approaches is that the vehicle's dynamics can be quite difficult to model. So called "first principle models" often are only an approximation to higher order phenomena. Even if the model's structure is accurate, estimating the underlying parameters can be quite difficult. For example, the rolling resistance coefficient is not constant, but rather a function of the terrain, tire tread wear, and temperature. As a result, nearly all model-based trajectory selection algorithms introduce a safety factor by dilating the predicted trajectories, prior to evaluation, in order to account for uncertainties in the future position of the vehicle. The dilation radius is frequently chosen in an *ad hoc* or heuristic fashion.

In this paper we present an enhancement of model-based trajectory selection algorithms which provides a principled and computationally efficient way to account for parametric uncertainty in the vehicle's dynamics using an error-controlled sparse grid interpolant. In this approach, the vehicle dynamics are simulated at model parameter values which are sparsely sampled from an estimated range of uncertainty. The simulations support the construction of a polynomial-based interpolant that describes the potential future trajectories of the vehicle subject to the simulated uncertainty.

The remainder of this paper is organized as follows. The following section presents a detailed review of the algorithmic structure of model-based trajectory planners, as well as approaches for dealing with parameter uncertainty. The

Method section describes how sparse grid interpolation techniques can be used to enhance such planners by accounting for this uncertainty in a principled and computationally efficient fashion by generating *trajectory clouds*. The *Model* section describes the differential drive vehicle model used to illustrate the approach. The *Implementation* section describes how the collision avoidance algorithm was modified and *Navigation Results* provides sample results. Finally, the *Discussion* section presents the advantages of this approach.

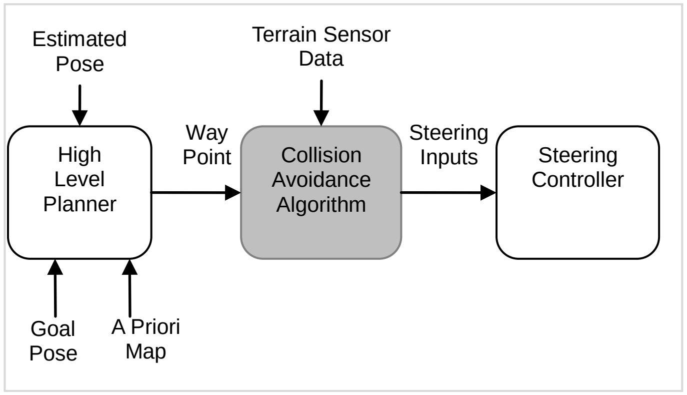

FIGURE 1. A COMMON MULTI-LEVEL MOTION PLANNING FRAMEWORK FOR AUTONOMOUS VEHICLES

LITERATURE REVIEW

There are many schemes for collision avoidance including variants on the potential field method [3], the dynamic window approach [4], vector field histograms [5], and graph search algorithms like D* [6]. However, these algorithms generate a desired velocity or input force which may not be directly implementable on a real vehicle. Non-holonomic kinematics, turning radius constraints, momentum, and actuator limits preclude certain instantaneous velocities. In the case of graph search algorithms the resulting trajectories can be also jagged and inefficient. Such algorithms require some type of post processing to produce feasible steering commands. In doing so, there is a risk that the collision avoidance guarantees of the original algorithm will be lost.

A very popular alternate approach is to collision check a local occupancy grid (a.k.a. ego-grid or cost-map) against a set of pre-computed trajectories generated directly from a discreet set of feasible inputs. A generic framework for such approaches is provided in Fig. 2 and illustrated in Fig. 3. Early implementations include [7] and [8], while more recent variants include [9] and [10]. They have been implemented extensively on ground vehicles in events such as DARPA Grand and Urban Challenges, and the European Land Robotics Trials (see for example [11-18]). More formally this is related to some of the Model Based Control concepts introduced in [11] and [12].

For each step in the algorithm shown in Fig. 2 there are many variations in the literature; however we focus on the precomputation steps 2 and 3 – the generation of the candidate trajectories and their dilation in such a way as to account for uncertainty. Dilation of the trajectory by a constant, albeit speed dependent, width is typical [7,8]. A more sophisticated approach would be to construct a probabilistic motion model such as that proposed in [20] Chapt. 5. However, the model used there affords a closed form solution and has only 4 uncertain parameters, making it difficult to extend to arbitrary motion models in a computationally efficient fashion. A general treatment of motion planning with model and sensing uncertainty requires the frame work of partially observable Markov decision processes (POMDPs), which is known to be intractable [21]. Approximations can render the approach viable in special cases, but simulating candidate trajectories is still required, similar to Fig. 2 (Precomputation Step 2 and 3).

```
Algorithm: Model-based Trajectory Selection
Pre-computation
    1. Create a discretized representation of the set of
       admissible inputs and state variables.
    2. Working in a body-fixed frame, generate trajectory
       segments corresponding to each of the inputs, at each
       of the states, from step 1.
    3. Dilate these segments as needed to account for the size
       of the vehicle and any additional clearance required to
       account for uncertainty.
    4. Set the resolution of the body fixed occupancy grid.
       Compute a lookup table indicating which cells support a
       given trajectory, include possible weighting coefficients
       to be used in path selection.

while (1)
    1. Determine desired direction of travel using a global
       planner
    2. Acquire sensor data describing surrounding terrain
    3. Populate body-fixed occupancy grid
    4. Evaluate each candidate trajectory segment for potential
       of collision or other fitness criteria (e.g. deviation from
       desired heading and speed)
    5. Execute steering inputs corresponding to the most
       desirable trajectory from Step 4, via the steering
       controller
end
```

FIGURE 2. A GENERIC OUTLINE FOR MODEL-BASED TRAJECTORY SELECTION ALGORITHMS

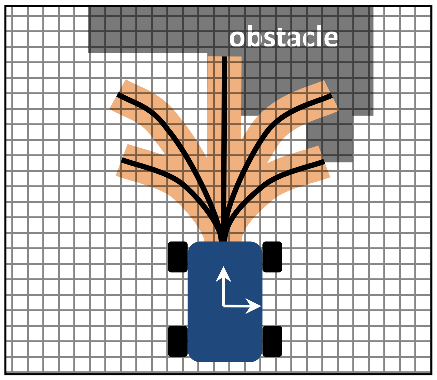

FIGURE 3. AN ILLUSTRATION OF MODEL-BASED TRAJECTORY SELECTION ALGORITHMS WITH AD HOC TRAJECTORY DILATION

METHOD

Consider Fig. 2: Precomputation Steps 2 and 3. Typically a candidate trajectory is computed based on the nominal vehicle dynamics as follows. Assume a vehicle has a dynamic model with states $x \in R^N$, outputs $y \in R^Q$, inputs $u \in U \subset R^M$, and model parameters $p \in P \subset R^D$. A state (or output) transition equation $y(t_k, u, x_o, p)$, which may be closed-form or evaluated via numerical simulation, can be used to predict the future output at time $t_k \in R^+$, assuming the initial state was $x_0 = x(t_0)$. Then an arbitrary safety margin is applied by taking the Minkowski sum of the outputs with a ball of radius r along a short but dense interval of times $T = [t_0, t_1, \cdots, t_K]$ (usually determined by the time steps of the simulation).

$$Y(T, u, x_0, p) = \cup_{k=0}^{K} y(t_k, u, x_0, p) \oplus B(r) \qquad (1)$$

Here Y represents the set of possible future positions that must be checked against the local occupancy grid for collisions and drivability.

We replace this *ad hoc* dilation with the concept of a trajectory cloud

$$Y(T, \bar{U}, \bar{X}_0, \bar{P}) = \cup\, y(t, u, x_0, p) \qquad (2)$$
$$\forall\ t \epsilon T, u \epsilon \bar{U}, x_0 \epsilon \bar{X}_0, p \epsilon \bar{P}$$

which reflects set-based uncertainty in the initial conditions, inputs and model parameters. Together we group these uncertain quantities into a single parameter vector $\theta\ \epsilon\ \bar{\Theta} = \bar{U} \times \bar{X}_0 \times \bar{P}$.

Clearly it is intractable to represent Y analytically for all but the simplest models, while computing it numerically would require an infinite number of simulations. Instead we generate an interpolation-based approximation of y, called $\tilde{y}(t, \theta)$, which is constructed from Lagrange polynomial basis functions. These interpolants are built upon evaluations of the state transition equation y at support nodes (t_k, θ_j) sampled using a sparse grid constructed on the set $\times\ \bar{\Theta}$ (see [22,23]). Interpolant generation is error-controlled, with support nodes added iteratively during grid construction to achieve a desired interpolation error with the minimum number of required simulations [23]. Given this closed form approximation (in interpolating polynomials) of the state transition equation, one can quickly form $\tilde{Y}$, an approximation of the set Y, by sampling the set $T \times \bar{\Theta}$ and evaluating the interpolant. Similar sparse grid interpolation approaches have been used to support model-based biological experiment design [24] and control [25].

A trajectory cloud is visualized in Fig. 4. The cloud illustrates all possible trajectories of a differential drive vehicle over a 3 second time period given a 10% uncertainty in the left and right wheel voltages, and initial forward velocity. To visualize the interpolant, points in the cloud were generated by Latin Hypercube Sampling (LHS) the parameter set, $T \times \bar{\Theta}$, and evaluating the interpolant at those points. This example is explained in more detail in the following section.

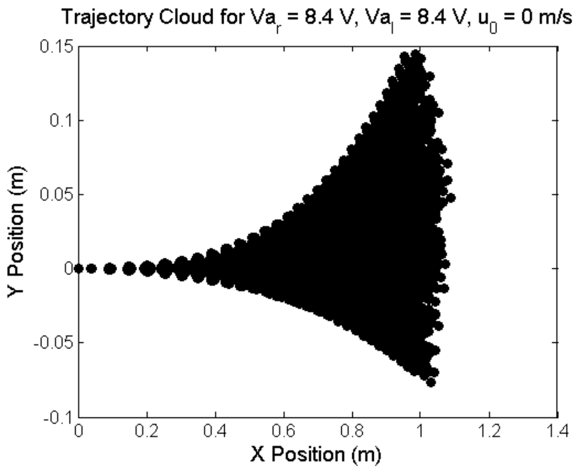

FIGURE 4. A SAMPLE TRAJECTORY CLOUD.

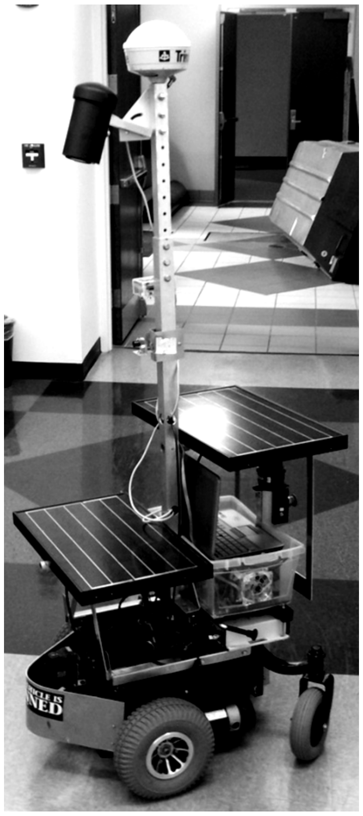

FIGURE 5. THE EXPERIMENTAL DIFFERENTIAL DRIVE VEHICLE

IMPLEMENTATION

We illustrate the use of trajectory clouds in collision avoidance on the differential drive vehicle shown in Fig. 5. The mathematical model used here was slightly modified from [26]. The 10 state variables consist of the position and orientation of the vehicle, $[x, y, \psi]$, the forward, lateral and angular velocities $[u, v, r]$, and the angular velocity and motor currents for the right and left wheels $[\omega_R, i_R, \omega_L, i_L]$. The inputs are the left and right wheel motor voltages $[V_R, V_L]$. The output is the position of the vehicle $[x, y]$. The model is provided below. There are an additional three equations required to compute the left wheel angular velocity, current, and force which are identical to the right wheel equations after replacing the subscript R with L.

$$\dot{x} = u\cos(\psi) - v\sin(\psi) \qquad (3)$$
$$\dot{y} = u\sin(\psi) + v\cos(\psi) \qquad (4)$$
$$\dot{\psi} = r \qquad (5)$$
$$\dot{u} = \frac{(F_R+F_L)}{m}\cdot\cos\left(-\tan^{-1}\left(\frac{v}{|u|}\right)\right) + vr - \frac{B_u}{m}u \qquad (6)$$
$$\dot{v} = \frac{(F_R+F_L)}{m}\cdot\sin\left(-\tan^{-1}\left(\frac{v}{|u|}\right)\right) - ur - \frac{B_v}{m}v \qquad (7)$$
$$\dot{r} = ((F_R - F_L)M_{arm} - B_r r)/J \qquad (8)$$

$$\dot{\omega}_R = (Ke_R i_R - B_\omega \omega_R - mg\sigma\cdot rad\,\omega_R)/J \qquad (9)$$
$$\dot{i}_R = (V_R - R_R i_R - Ke_R\omega_R)/L_R \qquad (10)$$
$$F_R = \left(\frac{V_R\cdot Ke_R}{rad\cdot R_R}\right) - \left(\frac{Ke_R^2}{rad\cdot R_R}\right)\omega_R \qquad (11)$$

The model parameters described in Tab. 1 were measured directly from the vehicle or fit to experimental data collected from the vehicle. The uncertain parameters on which the supporting sparse grid was constructed consisted of three model parameters, two inputs, and one initial condition.

$$\theta = [B_u, B_v, B_r, V_R, V_L, u_0]^T \ (D = 6). \qquad (12)$$

The parameters were assumed to vary within ±5% of their nominal value

$$\overline{\Theta} \equiv I_1 \otimes I_2 \otimes \cdots \otimes I_D \subset \mathbb{R}^D \qquad (13)$$

where $I_d = [0.95\theta^d_{nom}, 1.05\theta^d_{nom}]$, $d = 1, \ldots, D$, and θ^d_{nom} is the nominal value of the d^{th} uncertain quantity. For the first three parameters the nominal values are listed in Tab. 1, while the nominal values for the initial condition and inputs were the sensed and commanded quantities, respectively. Each model output was evaluated between $T = [0,3]$ sec. The interpolation error tolerances were set to max $(0.1|y|, 10^{-5})$. Constructing a typical interpolant for this model requires simulating the outputs at 90 support nodes (distinct vectors θ), where a single support node simulation requires about 0.025 sec to compute.

TABLE 1. MODEL PARAMETERS

	VALUE	UNITS	DESCRIPTION
Ke_R	3.11	NM/A	BACK EMF CONSTANT OR TORQUE CONSTANT FOR THE RIGHT MOTOR
m	81.8	KG	MASS OF THE VEHICLE
g	9.81	M/S^2	ACCELERATION DUE TO GRAVITY
rad	12.7	CM	RADIUS OF THE WHEELS
σ	0.02	N/A	WHEEL FRICTION COEFFICIENT
R_R	1.0	Ω	RESISTANCE OF RIGHT (OR LEFT) MOTOR
B_ω	2.19E^{-5}	N/A	WHEEL ROTATIONAL DAMPING COEFFICIENT
M_{arm}	25.4	CM	MOMENT ARM OF THE VEHICLE
L_R	0.01	H	INDUCTANCE OF RIGHT (OR LEFT) MOTOR
J	2.28	KGM^2	MOMENT OF INERTIA OF THE VEHICLE
B_u	8.1	KG	FORWARD DAMPING COEFFICIENT
B_v	100	KG	LATERAL DAMPING COEFFICIENT
B_r	1	KGM^2	ROTATIONAL DAMPING COEFFICIENT

The collision avoidance algorithm provided in Fig. 2 was implemented with some modifications. A set of trajectory clouds (describing different sensed initial conditions and commanded motor inputs) was precomputed to serve as a “look up table” describing potential paths of travel during vehicle navigation. The potential motor inputs were discretized such that
$V_R, V_L \in \{-4.8, -2.4, 0, 2.4, 3.6, 4.8, 6.0, 7.2, 8.4, 9.6, 10.8\}\,V$.
The set of 11 inputs for each motor means the vehicle can execute any combination of these values, allowing for 121 different input scenarios. Each of these 121 input scenarios was evaluated at 4 different forward velocities, $u_0 \in$

$\{0, 0.33, 0.66, 1\}$ m/sec. The result is a set of 484 trajectory clouds (interpolants) that describe vehicle paths for a variety of feasible vehicle states while accounting for uncertainty in the model characterization and controller execution.

Because the grid is constructed sparsely, with as few model evaluations as possible, the grid itself encodes the vehicle dynamics only at a few specifically sampled parameter vectors and time points. The space of the trajectory cloud is filled in by interpolating the model outputs at arbitrary parameter and time values within the sampled ranges. LHS was used to generate a set of 300 feasible vectors $\overline{\Theta}_f \subset \overline{\Theta}$ to evaluate $\tilde{Y}_j(T, \overline{\Theta}_f)$ for each of the j=1,…,484 interpolants, where T is a set of 30 uniformly distributed time points in the interval [0,3] sec. The trajectory clouds are populated with minimal computational expense as evaluating the interpolant only requires 0.001 sec of computation time.

Figure 6 shows a sample of trajectory clouds, where three different combinations of motor commands are plotted with varying initial velocity u_0.

After generation, the trajectory clouds are dilated to account for the width of the vehicle; the dilation is not used to address model uncertainty.

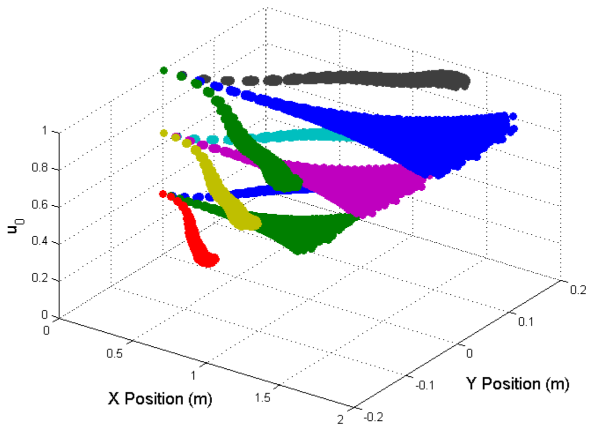

FIGURE 6. SAMPLE TRAJECTORY CLOUDS USED FOR VEHICLE NAVIGATION. WHILE ONLY 9 CLOUDS ARE PICTURED, THERE WERE 484 CLOUDS GENERATED TO SUPPORT THE NAVIGATION ALGORITHM.

NAVIGATION RESULTS

For our experiments the desired direction of travel is a constant compass bearing. Data from a LIDAR scan of the surrounding terrain (sampled at 10 Hz) is used to populate the body-fixed occupancy grid ($\Phi_{terrain}$) with a grid size of 6 m and a cell size of 5 cm.

At each sampling time, a sensor reading estimates current forward velocity u_0, which narrows the set of relevant trajectory clouds. Only the clouds constructed with u_0 closest to the measured value are retrieved and used for path selection. The remaining clouds now differ only in the combination of right and left motor voltage.

As in [4], the fitness of each remaining cloud is determined based on three metrics: deviation from desired speed, deviation from desired heading, and potential for collisions. The deviation from the desired speed

$$f_s = s_{des} - \bar{s}_{cloud} \tag{14}$$

computes the average speed of the cloud as

$$\bar{s}_{cloud} = \frac{\sqrt{\bar{x}^2_{T_{max}} + \bar{y}^2_{T_{max}}}}{T_{max}} \tag{15}$$

where $\bar{x}_{T_{max}}$ and $\bar{y}_{T_{max}}$ are the average x- and y-coordinates at the final time point, T_{max}, in the simulated trajectories. Here, the desired speed (s_{des}) was set to 0.5 m/s. The deviation from the desired heading

$$f_\psi = \psi_{des} - \psi_{cloud} \tag{16}$$

computes the heading of the cloud as

$$\psi_{cloud} = \tan^{-1}\left(\frac{\bar{y}_{T_{max}}}{\bar{x}_{T_{max}}}\right) \tag{17}$$

To bias the algorithm to drive straight, the desired heading (ψ_{des}) was set to 0.

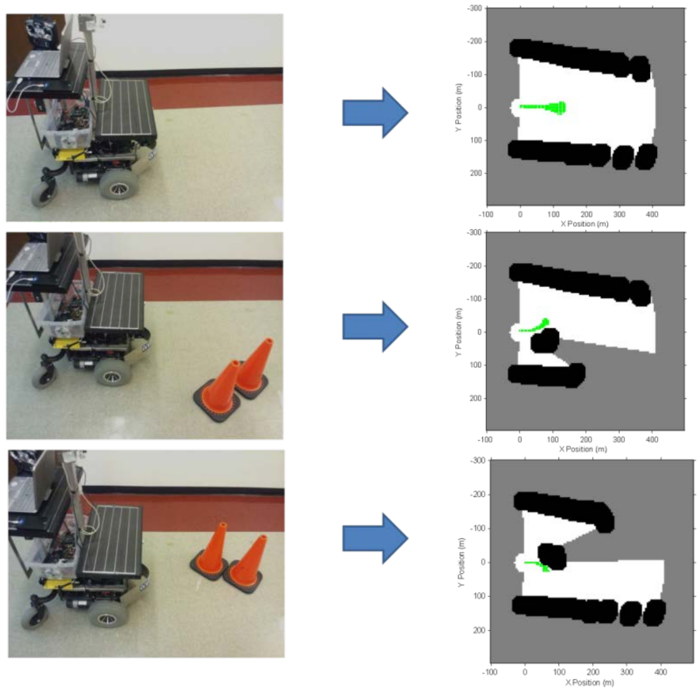

FIGURE 7. THREE FRAMES FROM THE HALLWAY NAVIGATION TEST

To evaluate the potential for collisions in a given trajectory cloud, the cloud is converted to an occupancy grid (Φ_{cloud}) identical to that generated for the terrain. The number of collisions, defined by the number of cells occupied by both an obstacle and the trajectory cloud, are summed for a given candidate cloud

$$f_c = \sum_i \sum_j c_{ij} \tag{18}$$

where

$$c_{ij} = \Phi_{\text{terrain}}(i,j) \wedge \Phi_{\text{cloud}}(i,j) \tag{19}$$

The overall cost function for the k^{th} trajectory cloud is the weighted sum of the three metrics

$$Cost^k = w_s f_s^k + w_\psi f_\psi^k + w_c f_c^k \tag{20}$$

where the weights were chosen as $w_s = 1, w_\psi = 0.01, w_c = 50$. The most desirable trajectory cloud is that with the lowest cost

$$\min_k Cost^k \tag{21}$$

and the motor voltages associated with this cloud are executed. This process repeats with the collection of updated sensor data.

Figure 7 shows three frames from a tests of the obstacle avoidance algorithm, the most desirable trajectory cloud is overlaid on the occupancy grid. During these experiments the robot was tasked with traveling along a compass heading that corresponded to moving down a straight hallway, approximately 20 meters long and 3 meters wide (the robot is 0.6 meters wide). In addition to the walls of the hallway, a set of 10 small traffic cones, 0.3 meters in diameter, were used to construct complex obstacle fields, including switchbacks, with passable gaps as narrow as 0.7 meters.

Clearly there is a trade-off between the amount of conservatism in specifying the range of the uncertain parameters and the ability to produce feasible trajectories that result in short paths. By setting the model parameter uncertainty interval to be small, resulting in thinner clouds, the robot was able to travel through gaps as small as 0.7 meters although the behavior was less repeatable. When the model parameter uncertainty intervals are assumed to be larger, the robot takes a more conservative yet repeatable path. Determining a realistic range of the uncertain model parameters is beyond the scope of this paper, yet it has a critical impact on the robot's behavior. Changing the ad hoc dilation radius of other model based collision avoidance algorithms has a similar effect.

More importantly, looking at Figure 7, the center and bottom panels illustrate scenarios in which the cloud based planner found a feasible solution, while an *ad hoc* dilation of the trajectory would not have found a solution. This is the major benefit of such an approach.

In our experiments the target speed was set to 2.23 m/s (5 miles per hour), which was the maximum speed of our platform. The planning algorithm updated at 10 Hz. This update rate was limited by the sensor package and not the collision avoidance algorithm, which averaged 0.02 sec per iteration on an HP Elite Laptop with an i5 Intel processor and 2 GB of RAM. In order to simulate the reactionary requirements required at higher speeds, we experimented with artificially limiting the range at which obstacles were detected. The Hokuyo URG laser range finder has a maximum detection range of 4 meters, but we achieved successful obstacle using a detection range as small as 1 meter (corresponding to a look-ahead time of 0.45 sec or 4.5 samples). Comparison with other algorithms is the subject of ongoing work.

DISCUSSION

The contribution of this paper is to introduce sparse grid sampling and interpolation as a way to account for parameter, initial state, and input uncertainty in model-based trajectory selection algorithms for autonomous vehicles. The vehicle dynamics are simulated at a sparsely sampled set of uncertain initial conditions, inputs, and model parameters. These simulations are used to construct an error-controlled interpolant approximating the set of future position of the vehicle for any arbitrary initial conditions, inputs, model parameters, and time points in the sampled spaces. Trajectory clouds of arbitrary density can be visualized by evaluating the interpolant. Evaluating the interpolant generally requires significantly less computation time than conducting further simulations.

There are several meaningful comparisons with alternate approaches. The traditional approach to dealing with model uncertainty, for collision avoidance, is to arbitrarily dilate the nominal trajectories, as depicted in Fig. 3. This approach suffers from two primary short comings. First of all, there is no accepted way to compute the dilation width. Second, the dilation width is uniform across time. Together this may result in overly conservative or aggressive input selections. In contrast the interpolant-based clouds shown in Fig. 4, 6 and 7 are generated by exploiting the model structure and the range of uncertain parameters. To the extent the model is structurally correct, these clouds represent a much higher fidelity representation of the uncertainty in the vehicle's final position. It is notable that the clouds are initially narrow, and become more uncertain over time. This is best illustrated by considering the trajectory clouds selected for execution in Fig. 7 (middle and bottom panels), which represent much more aggressive maneuvers than what would have been selected if a uniform dilation width, matching the final (3 sec) position uncertainty, was used.

A more principled alternative to uniform dilation would be to densely sample the parameter space and simulate the model in a brute force manner, either via a uniform grid on the parameter space or in a Monte Carlo fashion. While these would result in a high fidelity trajectory cloud, the approach presented here has three main advantages. First, it is error controlled to meet a specified relative and absolute tolerance.

Second, the sparse grid approach provides a computationally efficient way to capture the model output space, using as few simulations as possible. The number of required simulations is a function of the interpolation depth and error. The interpolation depth is the degree of the interpolating function while the error of the interpolating function strongly depends upon the smoothness and weakly depends on the dimension of the problem [27]. These methods are considered nearly optimal (up to a logarithmic factor) [27] and are significantly better than those of quasi-Monte Carlo algorithms [28]. Generally, once the interpolant is constructed, its evaluation is significantly faster than a numerical integration of the model (in our case, by an order of magnitude, likely more for stiff systems).

Finally, once constructed, the interpolant serves as a closed form approximation to the set of future positions of the vehicle. The interpolant can be re-evaluated to provide clouds of any density, and the evaluation can even be done adaptively if the resolution of the occupancy grid changes.

While we focused on model-based trajectory selection techniques for collision avoidance here, the trajectory cloud generation approach can find utility in other planning algorithms that require the simulation of trajectory primitives, such as the Rapidly Exploring Random Tree planner or model verification.

At present, our parameter interval uncertainty model is equivalent to assuming a uniform probability distribution on the uncertain parameter set. An area of future work is to associate an arbitrary probability distribution with each uncertain parameter. The sparse grid approach could then be used to construct a closed form approximation to the probability distribution of the vehicle's future positions. Such approximations may have utility in generating probabilistic motion models for use in mapping and localization applications as well as collision avoidance.

ACKNOWLEDGMENTS

The first two authors were supported by ONR grants.

REFERENCES

[1] Eds. Buehler, M., and Lagnemma, K., 2006. "Special Issue on the DARPA Grand Challenge". *Journal of Field Robotics*, 23(8 - 9).

[2] Eds. Buehler, M., Lagnemma, K. and Sinhg, S., 2008. "Special Issue on the DARPA Urban Challenge". *Journal of Field Robotics*, 25(8 - 10).

[3] Shimoda, S., Kurado, Y. and Iagnemma, K. 2005, "Potential field navigation of high speed unmanned ground vehicles on uneven terrain", in Proceedings of IEEE Conference on Robotics and Automation, p. 2828-2833.

[4] Fox, D., Burgard, W., and Thrun, S., 1997, "The dynamic window approach to collision avoidance," IEEE Robot Autom Mag, 4(1), pp. 23-33.

[5] Ulrich, I. and Borenstien, J., 2000, "VFH: local obstacle avoidance with look ahead verification", in Proceedings of IEEE Conference on Robotics and Automation, p. 2505-2511

[6] Koenig, S. and Likhachev, M., 2005, "Fast replanning for navigation in unknown terrain," IEEE Trans. On Robotics, 21(3), pp. 354-363.

[7] Lacaze, A., Moscovitz, Y., DeClaris, N., and Murphy, K., 1998, "Path planning for autonomous vehicles driving over rough terrain." Proceedings of the ISIC/CIRA/ISAS Conference, Gaithersburg, MD, September 14–17, 1998.

[8] Coombs, D., Murphy, K., Lacaze, A. and Legowik, S., 2000, "Driving autonomously off-road up to 35 km / h", in Proceedings of the IEEE Intelligent Vehicle Symposium, p. 186-191.

[9] Howard, T.M., and Kelly, A., 2007. "Optimal rough terrain trajectory generation for wheeled mobile robots," International Journal of Robotics Research, 26(2), 141–166

[10] von Hundelshausen, F., Himmelsbach, M., Hecker, F., Mueller, A., and Wuensche, H. J., 2008, "Driving with tentacles: Integral structures for sensing and motion," J Field Robot, 25(9), pp. 640-673.

[11] Dunlap, D.D., Caldwell, C.V. and Collins, E.G., 2010, "Sampling-based model predictive control", in Proceedings of the American Control Conference, Baltimore, MD p.240-245.

[12] Howard, T., Green, C., Ferguson, D., and Kelly, A.. 2008, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments", Journal of Field Robotics, 25(1):325–345.

[13] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y. W., Singh, S., Snider, J., Stentz, A., Whittaker, W. R., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., and Ferguson, D., 2008, "Autonomous driving in urban environments: Boss and the Urban Challenge," J Field Robot, 25(8), pp. 425-466.

[14] Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D., Cacciola, S., Currier, P., Dalton, A., Farmer, J., Hurdus, J., Kimmel, S., King, P., Taylor, A., Van Covern, D., and Webster, M., 2008, "Odin: Team

VictorTango's entry in the DARPA Urban Challenge," J Field Robot, 25(8), pp. 467-492.

[15] Miller, I., Campbell, M., Huttenlocher, D., Kline, F. R., Nathan, A., Lupashin, S., Catlin, J., Schimpf, B., Moran, P., Zych, N., Garcia, E., Kurdziel, M., and Fujishima, H., 2008, "Team Cornell's Skynet: Robust perception and planning in an urban environment," J Field Robot, 25(8), pp. 493-527.

[16] Patz, B. J., Papelis, Y., Pillat, R., Stein, G., and Harper, D., 2008, "A practical approach to robotic design for the DARPA Urban Challenge," J Field Robot, 25(8), pp. 528-566.

[17] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., and Thrun, S., 2008, "Junior: The Stanford entry in the Urban Challenge," J Field Robot, 25(9), pp. 569-597.

[18] Bohren, J., Foote, T., Keller, J., Kushleyev, A., Lee, D., Stewart, A., Vernaza, P., Derenick, J., Spletzer, J., and Satterfield, B., 2008, "Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge," J Field Robot, 25(9), pp. 598-614.

[19] Kammel, S., Ziegler, J., Pitzer, B., Werling, M., Gindele, T., Jagzent, D., Schroder, J., Thuy, M., Goebl, M., von Hundelshausen, F., Pink, O., Frese, C., and Stiller, C., 2008, "Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge," J Field Robot, 25(9), pp. 615-639.

[20] Rauskolb, F. W., Berger, K., Lipski, C., Magnor, M., Cornelsen, K., Effertz, J., Form, T., Graefe, F., Ohl, S., Schumacher, W., Wille, J. M., Hecker, P., Nothdurft, T., Doering, M., Homelier, K., Morgenroth, J., Wolf, L., Basarke, C., Berger, C., Gulke, T., Klose, F., and Rumpe, B., 2008, "Caroline: An autonomously driving vehicle for urban environments," J Field Robot, 25(9), pp. 674-724.

[21] Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D., Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M., Galejs, R., Krishnamurthy, S., and Williams, J., 2008, "A Perception-Driven Autonomous Urban Vehicle," J Field Robot, 25(10), pp. 727-774.

[22] Thrun, S., Burgard, W. and Fox, D., 2006. *Probablistic Robotics*, The MIT Press. Cambridge Massachusetts.

[23] Hauskrecht, M. 1997. "Incremental methods for computing bounds in partially observable Markov decision processes", in proceedings of the *AAAI National Conference in Articial Intelligence*. P 734-739.

[24] Bungartz, H. J., and Griebel, M., 2004, "Sparse Grids," Acta Numerica, 13, pp. 147-296.

[25] Klimke, A., and Wohlmuth, B., 2005, "Algorithm 847: spinterp: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB," Acm T Math Software, 31(4), pp. 561-579.

[26] Bazil, J. N., Buzzard, G. T., and Rundell, A. E., 2011, "Minimizing Model Output Uncertainty Using a Global, Parallel Model-Based Design of Experiments Approach," Faseb J, 25.

[27] Noble, S. L., Wendel, L. E., Donahue, M. M., Buzzard, G. T., and Rundell, A. E., 2012, "Sparse-Grid-Based Adaptive Model Predictive Control of HL60 Cellular Differentiation," IEEE Transactions on Biomedical Engineering, 59(2), pp. 456-463.

[28] Worrall, K., and Mcgookin, E., 2006, "A mathematical model of a lego differential drive robot," International Control ConferenceGlasgow, Scotland.

[29] Barthelmann, V., Novak, E., and Ritter, K., 2000, "High dimensional polynomial interpolation on sparse grids " Advances in Computational Mathematics, 12(4), pp. 213-288.

[30] Gerstner, T., and Griebel, M., 2003, "Dimension-adaptive tensor-product quadrature," Computing, 71(1), pp. 65-87.