2017 IEEE Intelligent Vehicles Symposium (IV)
June 11-14, 2017, Redondo Beach, CA, USA

Systematically Comparing Control Approaches in the Presence of Actuator Errors

Tobias Kessler[1], Pascal Minnerup[1], David Lenz[1], Alois Knoll[2]

***Abstract*— A quantitative comparison of different approaches to control the trajectory of an autonomous vehicle can be achieved by measuring the overall performance of the whole system in closed loop. This work compares two trajectory generation and control approaches for autonomous driving in simulation. One approach is based on a low frequency check of geometric safety distances and a high frequency check of control deviations. The other approach minimizes the probability of colliding with obstacles by explicitly considering actuator uncertainties and solving a chance constraint optimization problem. The comparison is based on systematically applying actuator inaccuracies and searching for event combinations leading to collisions. We demonstrate that the control approach incorporating uncertainties performs better in one class of critical scenarios although its error model is different to the applied inaccuracies.**

I. Introduction

Evaluating the performance of a complex cyber physical system is a challenging task and a fair comparison of different approaches requires additional effort. As software in vehicles is gaining importance, so is this effort for benchmarking different algorithms. Choosing the most suitable approach for a specific (driving) problem is essential in an early development phase. The evaluation criteria are numerous and a suitable set of measurable performance indicators has to be extracted. The main part of the evaluation focusing on the software parts of the cyber physical system can be conducted in simulation, if sufficiently accurate models are available.

Two aspects need to be considered when evaluating the performance of a planning and control system: The average and the worst case performance. A Monte Carlo simulation can be used for evaluating the average performance. The worst case performance corresponds to the ability of the planning and control system to cope with unfavorable combinations of sensor and actuator inaccuracies. In [1], a method has been introduced to evaluate this worst case performance efficiently. It directly tests the implementation of the planning and control system in the presence of complex combinations of inaccuracies including delays. In this paper, this testing algorithm is applied to directly compare two different implementations of a planning and control system. This extends the application of the test algorithm and demonstrates its applicability to different software implementations.

One challenge in the motion planning and control of an autonomous vehicle is how uncertain sensor measurements

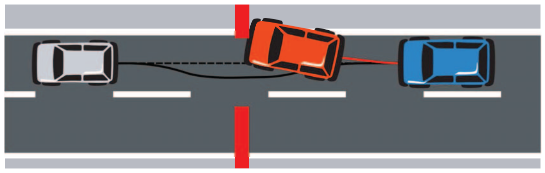


Fig. 1. Schematic sketch of discussed problem setup. A vehicle shall maneuver safely from the start position (blue, right) to the goal position further on the road (gray, left) passing a tunnel of obstacles (red) with varying actuator errors. A suboptimal error combination can result in a colliding trajectory (red).

and non-accurate actuator commands are integrated into the planning and control process. With a conservative approximation, safety margins can be defined, that yield a safe (collision free) motion but limit the performance of the system. On the other hand, uncertainties can be directly integrated into the control process. In this case, reasonable probability distributions have to be measured or estimated.

In this work, we focus on the comparison of different control algorithms in an autonomous vehicle within two different scenarios. The algorithms are a model predictive controller incorporating uncertainties [2] and one polynomial trajectory generator with a state feedback controller relying on moderate safety distances [3]. By extensive simulation, we evaluate the safety of both approaches as we try to find scenarios that result in a collision. The model predictive controller is computationally more expensive and it is desirable that these higher runtime costs result in a safer motion. The observed problem is sketched in Fig. 1. The start position (blue, right) is connected to the goal position (gray, left) with a collision free path (black, dashed) that does not pass the tunnel right in the middle. Both trajectory control approaches tend to take a safe trajectory deviating from the reference path right in the middle of the tunnel (black, solid). A bad combination of actuator errors can result in a trajectory (red, solid) resulting in a collision (red). The here discussed algorithms differ in how they deal with the occurring errors and how huge errors can be compensated by effective controls. In the following we will

- give a measurable comparison of different vehicle control approaches dealing with sensor and actuator uncertainties,
- analyze how a chance constraint stochastic model predictive controller can resolve a driving scenario with worst case error patterns,
- show how two different vehicle control approaches can be evaluated using the *STARVEC* testing framework.

[1]Tobias Kessler, David Lenz, and Pascal Minnerup are with fortiss GmbH, An-Institut Technische Universität München, Munich, Germany
[2]Alois Knoll is with Robotics and Embedded Systems, Technische Universität München, Munich, Germany

This work is further organized as follows. Section II gives an overview of methods for systematically testing planning and control systems. In section III we describe the *STARVEC* test framework and the compared vehicle control approaches in detail. Furthermore the system parameters, especially regarding the applied error models are given. The simulation results of the test scenario are given in section IV and are followed by a conclusion in section V.

II. Related Work

There are three approaches for comparing planning and control systems in a simulation environment. One is to create a simulation that is as realistic as possible such that evaluation in simulation is similar to evaluation in reality. The second approach is to automatically generate different situations in which the planning and control system can be tested. The third approach is to model possible events and test the system in the presence of these events.

Realistic simulation has been considered of major importance by the teams of the DARPA urban challenge. For example, the Tartan racing team [4] describes that they first tested their algorithms in a simulation environment. In order to increase the realism of the simulation, exact sensor models can be used. For this purpose, the authors of [5] work on improving such models. For maximizing the realism of vehicle dynamics, the VEHIL setup [6] uses an actually moving base for representing the vehicle dynamics.

The second approach is to automatically generate different situations for testing the planning and control system. The authors of [7] test a parking system in a simulation environment. In order to find situations in which the planning and control system fails, they developed a metric to measure how close the system is to failure. Based on this measure, they apply genetic algorithms for modifying the scenario until they find one in which the system under test fails. As an alternative, [8] proposes to collect simulation scenarios while performing physical test drives. If a scenario seems to be challenging for the physical vehicle, it is stored for offline analysis in a simulation environment.

The third approach is to model events and test the system against combinations of these events. In [9], model checking is extended with dynamic analysis. For this purpose, the model checking tool focuses on an abstraction of the actual software state, but the actual software including all states is executed in between. A similar concept is used by the *STARVEC* algorithm [1] that focuses on covering the geometric space, but executes the actual planning and control software to get from one state to another. The authors of [10] compare the performance of two controllers with some events affecting them. They use rapidly exploring random trees to reach different states efficiently. In contrast to that paper, the present paper and the *STARVEC* algorithm test software systems with more complex states and more complex error patterns including delays.

In this work, two trajectory generation and vehicle control approaches are compared, one incorporating actuator uncertainties using chance constraints on the system states. The concept of chance constraints is discussed in e.g. [11] and does not enforce exact constraint satisfaction but guarantees to limit constraint violations up to a certain probability. Vitus and Tomlin [12] show how a chance constraint framework can outperform approaches relying on conservative safety bounds. A recent discussion on the topic can be found in [13]. This work uses the control approach introduced in [2].

III. System Setup

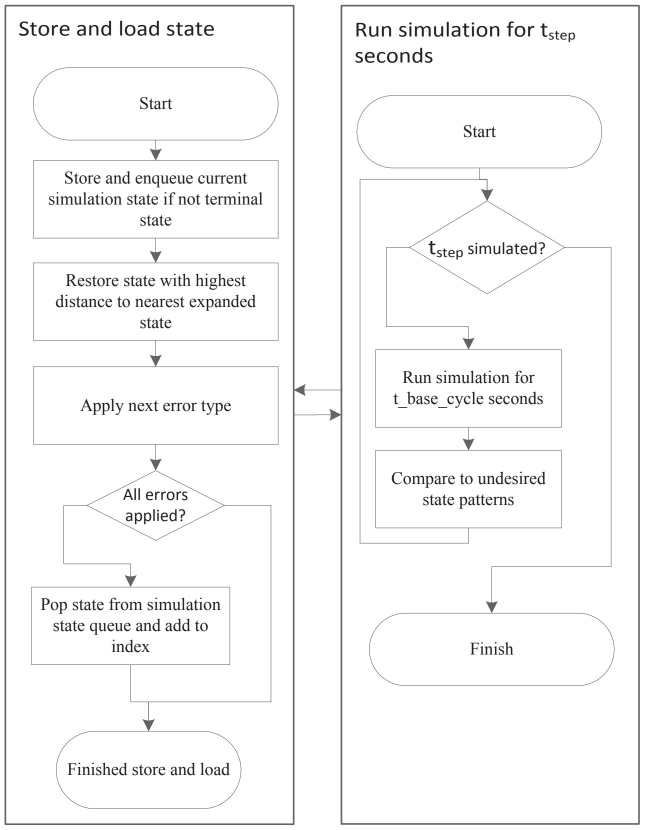
This section introduces the testing framework and the planning and simulation environment. Both compared vehicle control approaches are described with a focus on their differences in handling driving situations involving obstacles. The planning and control framework is capable of complex driving scenarios, cf. [14]. Only a subset of the functionality is applied here, as the test scenarios are kept simple.

We compare two vehicle control approaches, both with different strategies to handle sensor and actuator errors. The first as introduced in section III-C generates a trajectory fan and selects the best among those. Collisions are avoided by setting a (conservative) safety distance a trajectory has to keep to obstacles. The latter chance constraint approach in section III-D assumes a previously known probability distribution of actuator errors and aims to avoid collisions up to a certain probability. Clearly, both approaches have to be parametrized in a comparable manner.

A. Testing Framework

The *STARVEC* framework [1] is used for comparing the performance of the two competing planning and control implementations. Based on the capability to store and restore the state of the entire trajectory planning component, it can efficiently cover the space of reachable states. Fig. 2 shows an overview over the test algorithm. It starts with storing the current state of the system under test and the simulation environment. This state is enqueued in a special priority queue. The priority queue maintains the stored states and their distance to the next fully expanded states. After enqueuing the simulation state, the state with the maximal distance in the queue is restored and a new error pattern is applied to the simulation. If all different error patterns have been applied to the loaded state, it is removed from the priority queue, else it can be loaded again with another error pattern. Next, the simulation is executed starting with the loaded state. During the execution, the applied error pattern affects the behavior of the vehicle. After some time, the simulation is suspended for repeating the described store and load. In parallel to the simulation, the *STARVEC* framework continuously checks, if an undesired behavior like a collision has been performed by the system under test.

The effect of the *STARVEC* algorithm is shown in Fig. 3. Each step, multiple possible behaviors of the system under test induced by different error patterns are simulated. This leads to a tree of possible simulation states. The test algorithm executes states that are very different. This way the tree quickly spans a large area potentially including positions at which the vehicle collides with static obstacles.



Store and load state
Start
Store and enqueue current simulation state if not terminal state
Restore state with highest distance to nearest expanded state
Apply next error type
All errors applied?
Pop state from simulation state queue and add to index
Finished store and load
Run simulation for t_{step} seconds
Start
t_{step} simulated?
Run simulation for t_base_cycle seconds
Compare to undesired state patterns
Finish

Fig. 2. Overview of the *STARVEC* algorithm for finding error pattern combinations leading to undesired states. Loading states and applying new inaccuracy patterns alternates with executing a short part of the simulation sequence.

B. Planning and Simulation Environment

We simulate the vehicle as single track model and apply offsets on the current vehicle acceleration and steering wheel angle as well as varying acceleration command and steering command delay.

A polynomial representation of the environment is used. Each obstacle is represented as a set of vertices on the ground plane. For the sake of simplicity, the sensed environment is assumed static and does not change while the vehicle is moving.

To simplify the experimental setup we do not compute a driving strategy but use a fixed start and goal pose for the vehicle under test. The reference path is generated using a variant of the Hybrid A* search algorithm with post optimization as introduced in [15]. The reference path is collision free, curvature continuous, and sampled with a certain point distance. As we aim to examine critical situations for the vehicle controller, we set the safety distance the path planner keeps to obstacles to $0.1m$. The reference path is computed before the evaluations are started and does not change. For the trajectory generation the same algorithm is used for both of the compared control algorithms but with different parametrization as detailed in the further sections. The target vehicle speed is set to $1m/s$.

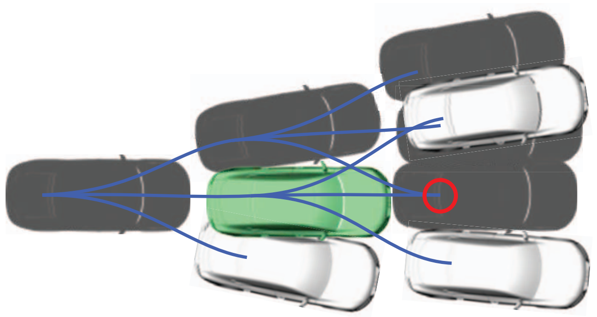


Fig. 3. Visualization of the *STARVEC* algorithm: Each intermediate state is loaded several times for exploring alternative possible future paths (blue lines). If two states are similar, (paths meet in the red circle) only one of them is explored.

C. Polynomial Trajectory Generation and Control

The implemented trajectory planning and control approach by Werling and Gröll [3] and Werling *et al.* [16] generates a set of trajectories along the reference path in a local frame. From these the collision free jerk optimal one is selected and passed to a controller. The Lyapunov-based control law tracks the current trajectory point in time and does not consider obstacles any more as these are assumed to be accounted in the trajectory generation. The whole process is not aware of sensor or actuator uncertainties. All are assumed to be covered by a trajectory planner safety distance of $0.2m$. We generate trajectories with a length of 12 seconds and sample trajectories with a maximum lateral distance to the reference path of $0.1m$. With a significantly higher safety distance, no trajectory that can pass the evaluated scenarios can be found and the vehicle stops in front of the critical situation. This obviously is a safe motion but also a bad overall performance. The trajectory is recalculated every $0.5s$. With controller errors lower than a threshold of $1m$ the new trajectory is a continuous continuation of the old trajectory. Otherwise, a new trajectory starting from the current vehicle state is generated. The controller interpolates the current point in time on the trajectory and aims to minimize the trajectory tracking errors with respect to longitudinal motion, lateral motion, orientation, and velocity. We will refer to this approach as *State Feedback Controller* in the following.

D. Chance Constraint Stochastic Model Predictive Control

The design of the Chance Constraint Model Predictive Controller (referred to as *CC-SMPC*) is introduced in [2]. This work aims at a control strategy that overcomes the parametrization effort of a Model Predictive Control approach to satisfy both, the need for safe (collision free) behavior and comfortable (smooth) trajectory following. As constraints are taken into account in the algorithm construction, the controller is safe by design and in the parametrization phase, only comfort issues have to be addressed.

With the polynomial trajectory generation approach described in section III-C, we generate exactly one trajectory along the reference path with a smooth acceleration and deceleration profile. As the reference path is collision free and the obstacles shall be accounted in the trajectory control we set the safety distance to zero. The trajectory is generated with a horizon of $12s$. The trajectory is regenerated every $0.2s$. The horizon of the model predictive controller are $2s$.

The constraints along the planned trajectory are generated from the static obstacle map in the planning layer. The constraints are generated to be convex and linear but time dependent. Compared to the optimization/control step the constraint generation takes a negligible amount of time. Based on an approximation of the car shape by a number of circles we create a set of constraining hyperplanes. The circle diameter is chosen to exactly fit the vehicle with and do not introduce an additional safety margin here. We approximate the vehicle using two circles. Taking the full set of kinematic constraints (x, y, θ) into account and applying the small angle approximation along the calculated reference, we construct a set of linear convex constraints.

The controller aims to minimize a quadratic objective function $J(\boldsymbol{u}, \boldsymbol{x})$ with the system states $\boldsymbol{x} = \begin{pmatrix} x, & y, & \theta, & v \end{pmatrix}$ and control inputs $\boldsymbol{u} = \begin{pmatrix} a, & \sigma \end{pmatrix}$ with steering curvature σ and acceleration a. As vehicle model, we chose a (non-holonomic) single track model with acceleration and steering curvature as inputs and position x, y, heading θ, and velocity v as states. Obstacles impose state constraints. As we require the state constraints to be fulfilled with a certain probability, we get a chance constraint system. The chance constraints are transformed to ordinary inequality constraints and reformulated in terms of the input variables. Using a block matrix (condensed) formulation the problem can be written in standard form and solved with an off-the-shelf solver. The optimization problem is set up to directly output the vehicle control vector $\boldsymbol{u}$ that is passed to the vehicle dynamics simulation. The control vector is bounded by actuator limits and are assumed deterministic. As the model predictive controller runs at a slower rate, the control vector is appropriately interpolated in the horizon.

Note that using a chance constraint framework the constraints are only satisfied up to a certain probability α that we chose as 0.95 here. Due to the covariance propagation in the time steps, constraints closer to the vehicle tend to be fulfilled whereas constraints in the future are violated with a higher probability.

The control algorithm is implemented in C++ using the NLopt[1] nonlinear optimization library to solve the optimization problem. As an optimization solver, we use the SQP method [17]. As bound constraints for the optimization, we chose the physical limits in the steering angle and acceleration, deceleration of the vehicle model.

E. Error Modeling

The *STARVEC* framework and the Chance Constraint Model Predictive Controller use an entirely different error model. *STARVEC* samples errors in a uniform distribution whereas the CC-SMPC relies on a Gaussian distribution of errors. As the simulation is controlled by the *STARVEC* test framework and the error patterns are directly applied to the vehicle simulation, this error model can be treated as ground truth. The assumed probability distribution of the controller has to (conservatively) model the *STARVEC* error pattern.

[1]http://ab-initio.mit.edu/nlopt (Steven G. Johnson)

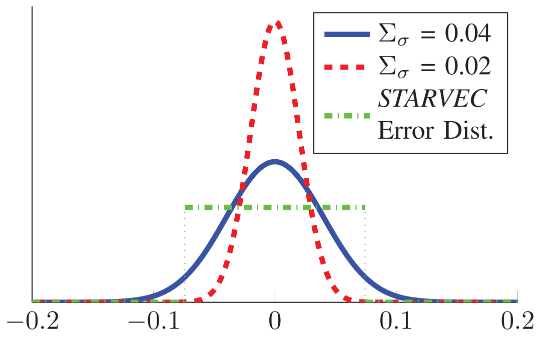


Fig. 4. Gaussian curvature distributions used by the CC-SMPC and uniform distribution including offset and delay error patterns used by the *STARVEC* framework

Using a model-predictive control approach a certain error characteristic for the actuator errors has to be assumed. We chose a the process noise to be a normal distribution $\mathcal{N}(0, \boldsymbol{w})$ with zero mean and covariance $\boldsymbol{\Sigma_w}$ as

$$\boldsymbol{\Sigma_w} = \begin{bmatrix} \Sigma_a & 0 \\ 0 & \Sigma_\sigma \end{bmatrix}.$$

We do not apply a state covariance for the initial state $\boldsymbol{x}_0$ of the controller. We chose $\Sigma_a = 0.5$ and vary Σ_σ between 0.02 and 0.04.

The analysis framework *STARVEC* instead samples errors in a uniform distribution. Two different error patterns are applied: A constant offset to the controller command values and a delay leading to old control values being applied instead of the current control commands. We set the maximum steering command offset to $0.07rad$ and the maximum acceleration offset to $0.05m/s^2$. Both maximum delay times are set to $0.2s$.

The difference in the probability distributions for both error models can be motivated graphically. Fig. 4 shows the curvature error distribution comparing the normal distribution assumed by the controller and the uniform distribution used by the test framework. As the delay cannot directly be expressed in the distribution, we increase the offset as follows. Assuming a maximum possible steering rate for the vehicle model for the full maximum delay time we get a worst case additional curvature offset. This we add to maximum offset. The figure shows the curves of the Gaussian error model of the CC-SMPC and the uniform distribution the *STARVEC* framework draws the error from, including both offset and delay errors. Varying the covariance values used by the controller can introduce conservatism or aggressiveness into the system. The probability distribution for the acceleration profile is comparable.

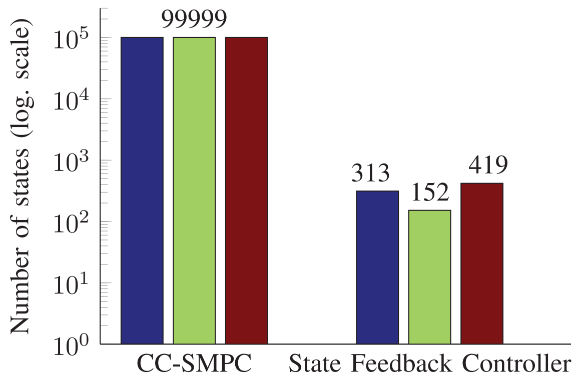


Fig. 5. Comparison how many states were explored by the *STARVEC* algorithm on a logarithmic scale. The evaluation is terminated if a collision was found.

IV. Evaluation

We will describe the test scenarios and the criteria for evaluating the simulation results. In this work, we do not focus on a comprehensive evaluation of a multitude of possible scenarios but on the evaluation, how incorporating uncertainties in the controller design can effect critical situations, detected by the *STARVEC* testing approach.

A. Simple Benchmark Scenario

As a benchmark scenario, we apply the simple tunnel scenario as depicted in Fig. 1. Without sensor and actuator errors, the planed path (black, dashed) is safe (collision free). Both, a trajectory generation and control approach relying on conservative safety distances as described in section III-C and the chance constraint control approach (see section III-D) chose a trajectory maximizing the safety margins in the critical tunnel section (black, solid). The simulation is stopped if either a collision is found or a maximum number of steps is reached. For the CC-SMPC, we chose $\Sigma_\sigma = 0.02$.

The main intention of the *STARVEC* algorithm is finding an error pattern yielding a colliding state. We ran the identical scenario three times for both control approaches. We found that the state feedback control approach in this parametrization is very likely to collide after a few hundred steps as shown in Fig. 5. If the safety distance is increased, the tunnel cannot be passed any more within this higher safety margin. With the CC-SMPC controller, we do not find any colliding state but terminate the evaluation after the indicated number of states.

STARVEC aims to find colliding states and tends to explore critical regions earlier. The total area covered by the vehicle shape at all instances is an indicator how a controller reacted to the error pattern and how it tried to keep the vehicle out of critical situations with immanent obstacle collisions. Fig. 6 and Fig. 7 show this covered areas (gray) and the saved vehicle states (figures show the rear axle center xy-position of the vehicle). The *STARVEC* algorithm applies error patterns that keep the vehicle close to the right hand obstacle and eventually finds a collision here for the state feedback control approach. In all three runs, the colliding state is roughly

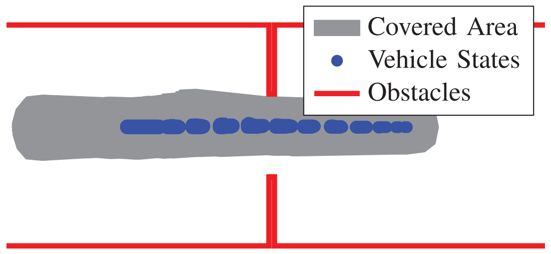


Fig. 6. Vehicle states and covered area by all CC-SMPC controller simulations in the tunnel scenario. No collision is found and a multitude of vehicle states is explored.

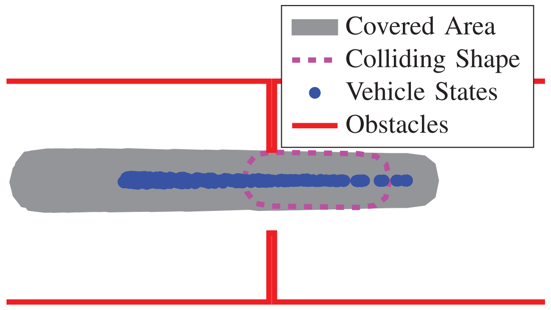


Fig. 7. Vehicle states and covered area by all state feedback controller simulations in the tunnel scenario. The dashed magenta vehicle shape shows one of the colliding positions.

at the same position. The CC-SMPC approach is able to compensate the error patterns and to use the available free space to the left of the vehicle to avoid a collision. The figures show all states reached in all three simulations runs.

B. Parking Scenario

We further evaluate the performance of both control approaches in a parking-like scenario. The scenario consists of two direction changes and the reference path also takes curves. The goal pose is inside a tunnel shifted to the left. For the State Feedback Controller we lower the safety distance to $0.15m$, for the CC-SMPC controller we lower the curvature covariance to $\Sigma_\sigma = 0.04$. Again, with a significantly higher safety distance the tunnel cannot be entered any more. Otherwise, the parameters are unchanged compared to the simple scenario section IV-A. The scenario as well as the results are depicted in Fig. 8. The covered area is similar for both control approaches, in the CC-SMPC case more area is explored due to a higher number of expanded states. Table I shows the number of states needed to find a colliding state for the state feedback controller. With the CC-SMPC controller a collision is never found but the simulation is terminated after the indicated number of runs.

TABLE I
Number of states the simulation is terminated

	State Feedback Controller (collision found)	CC-SMPC (simulation aborted)
run 1	1498	21216
run 2	2407	21270
run 3	378	21283

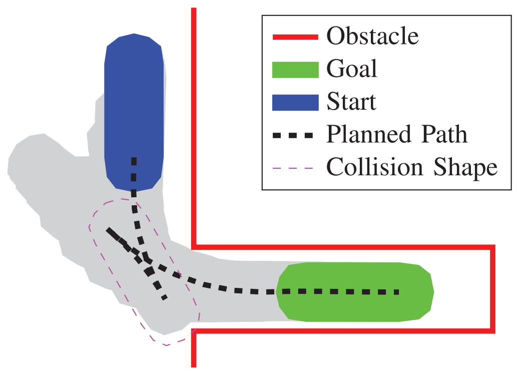


Fig. 8. Scene evolution and covered space (gray) by the planned path in the parking scenario. The colliding shape shows one of the found collisions.

V. Conclusion

We have shown how a control approach that takes into account sensor and actuator uncertainties can outperform an approach that relies on conservative safety margins in two scenarios. These are chosen to be challenging for both control approaches. The performance of both can be increased by parameter tuning. This is not essential in case of the CC-SMPC control approach and underlines that the approach is safe by design and has to be tuned mainly for comfort reasons. Never the less this is not a comprehensive evaluation of both approaches and no conclusion can be drawn that CC-SMPC always outperforms the state feedback controller.

The *STARVEC* test framework was used to find critical situations with both control approaches. It is capable of quantitatively comparing the safety of two different control approaches. A more general answer on the performance of both control approaches shall be achieved by simulating more sophisticated scenarios. This does not change the methodology of the testing approach.

Furthermore, the parametrization of the CC-SMPC approach and the *STARVEC* error pattern are chosen to be close to realistic values but have not been validated which is planned as future research.

References

[1] P. Minnerup and A. Knoll, "Testing Automated Vehicles against Actuator Inaccuracies in a Large State Space," in *9th IFAC Symposium on Intelligent Autonomous Vehicles*, Leipzig, Germany, 2016.

[2] D. Lenz, T. Kessler, and A. Knoll, "Stochastic Model Predictive Controller with Chance Constraints for Comfortable and Safe Driving Behavior of Autonomous Vehicles," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015.

[3] M. Werling and L. Gröll, "From flatness-based trajectory tracking to path following," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 1271–1275, 2009.

[4] C. Urmson, J. Anhalt, D. Bagnell, *et al.*, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[5] E. Roth, T. J. Dirndorfer, A. Knoll, *et al.*, "Analysis and Validation of Perception Sensor Models in an Integrated Vehicle and Environment Simulation," in *22nd International Technical Conference on the Enhanced Safety of Vehicles (ESV)*, Washington, DC, USA, 2011.

[6] L. Verhoeff, D. Verburg, H. Lupker, *et al.*, "VEHIL: a full-scale test methodology for intelligent transport systems, vehicles and subsystems," in *IEEE Intelligent Vehicles Symposium (IV)*, Dearborn, USA: IEEE, 2000, pp. 369–375.

[7] O. Buhler and J. Wegener, "Automatic testing of an autonomous parking system using evolutionary computation," *SOCIETY OF AUTOMOTIVE ENGINEERS INC.*, pp. 115–122, 2004.

[8] P. Minnerup, T. Kessler, and A. Knoll, "Collecting Simulation Scenarios by Analyzing Physical Test Drives," in *18th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Las Palmas de Gran Canaria, 2015, pp. 2915–2920.

[9] A. Groce and R. Joshi, "Extending Model Checking with Dynamic Analysis," in, ser. Lecture Notes in Computer Science, F. Logozzo, D. A. Peled, and L. D. Zuck, Eds., Springer Berlin Heidelberg, Jan. 2008, pp. 142–156.

[10] D. Hes, M. Althoff, and T. Sattel, "Comparison of trajectory tracking controllers for emergency situations," in *IEEE Intelligent Vehicles Symposium, Proceedings*, IEEE, 2013, pp. 163–170.

[11] P. Kall and J. Mayer, *Stochastic Linear Programming Models, Theory, and Computation*, 2nd ed. Springer, 2011.

[12] M. P. Vitus and C. J. Tomlin, "A probabilistic approach to planning and control in autonomous urban driving," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 2459–2464.

[13] M. P. Vitus, Z. Zhou, and C. J. Tomlin, "Stochastic control with uncertain parameters via chance constrained control," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2892–2905, 2016.

[14] D. Lenz, P. Minnerup, C. Chen, *et al.*, "Mehrstufiges Planungskonzept fuer pilotierte Parkhausfunktionen," in *30. VDI/VW-Gemeinschaftstagung "Fahrerassistenz und Integrierte Sicherheit 2014"*, Wolfsburg, Germany, 2014.

[15] D. Dolgov, S. Thrun, M. Montemerlo, *et al.*, "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[16] M. Werling, L. Gröll, and G. Bretthauer, "Invariant trajectory tracking with a full-size autonomous road vehicle," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 758–765, 2010.

[17] D. Kraft, "Algorithm 733: TOMP-Fortran modules for optimal control calculations," *ACM Transactions on Mathematical Software*, vol. 20, no. 3, pp. 262–281, 1994.