2018 Annual American Control Conference (ACC)
June 27–29, 2018. Wisconsin Center, Milwaukee, USA

Safety Verification Of ADAS By Collision-free Boundary Searching Of A Parameterized Catalog

Jinwei Zhou[1], Luigi del Re[1]

***Abstract*— Safety evaluation and verification are challenging for the development of the Advanced Driver Assistance Systems(ADAS) or Automated Driving Functions(ADF). Real world road testing requires an unaffordable testing time. Therefore, simulations are included so that ADAS/ADF are tested and verified in a virtual environment. In the previous work, a test-case catalog has been introduced to limit the total number of test cases. Realistic parameterization of test-cases based on experimental data allows safety assessment through the estimation of unsafe conditions (*e.g.* collision rate) in terms of the real world traffic situation, by looking for the safety boundary, separating safe conditions from unsafe. Due to the high dimensionality of complex test cases, a rigorous grid searching is highly time consuming.**

Against this background, we propose an improved Input Design method based on Gaussian Process Classification algorithm to handle the safety boundary searching problem. This method allows an efficient identification of non-convex boundaries. It guarantees a good approximation of the true boundary based on limited number of simulations (or measurements).

I. INTRODUCTION

In the last years, great effort has been made to increase the automation level in vehicles to handle more complex traffic situations. For example, an automated driving system for highway traffic, consisting of a full-speed range adaptive cruise control (FSR-ACC) and a lane keeping system (LKS), should be able to handle the operational and tactical aspects of the driving task, such as cruising in the current driving lane, keeping a safe distance from the leading vehicle and switching the leading vehicle in time in the case that a vehicle in the adjacent lane cuts in or the leading vehicle pulls out. Before the diffusion of these automation systems, intensive testing is mandatory to verify their safety. A real world field test requires an unaffordable amount of kilometers of test driving [1]. Therefore, simulation based (virtual) testing must be included. In virtual testing, the deterministic methods, *e.g.* [2]–[4], and the stochastic modeling methods, *e.g.* [5], are employed to produce test scenarios. However, the deterministic methods only focus on the predefined use cases or limited critical events, while stochastic methods have low exposure to critical events. Furthermore, these method are based on a simplification of real traffic situations, making it difficult to evaluate the safety of the ADAS/ADFs in the real world use.

To overcome this limitation, a test case catalog, derived from the traffic accident database, was introduced to limit the number of total test cases [6]. A suitable parameterization of a test case, using experimental data, allows the determination of a safety boundary, separating safe conditions from unsafe, and the estimation of the probability of these conditions in real world traffic situations. This allows assessing the safety of the system under test (SUT), *e.g.* the collision rate. However, an efficient identification of the safety boundary is a challenging task. Due to the system complexity, it is hard to achieve an exact analytical solution. In [6], a grid searching was applied to determine the boundary numerically, which is highly time consuming in the case of high input dimensionality or high precision. To accelerate the process of boundary searching, a variety of design of experiment (DoE) strategies is available. The support vector machines (SVM) [7] and the Gaussian Process Classification (GPC) [8] based Input Design method are suitable algorithms for searching and describing the non-convex boundaries, see Fig. 1. However, both algorithms are based on the predefined iteration steps. Thus, the approximation quality is unknown and the corresponding exit condition of the iteration is not given.

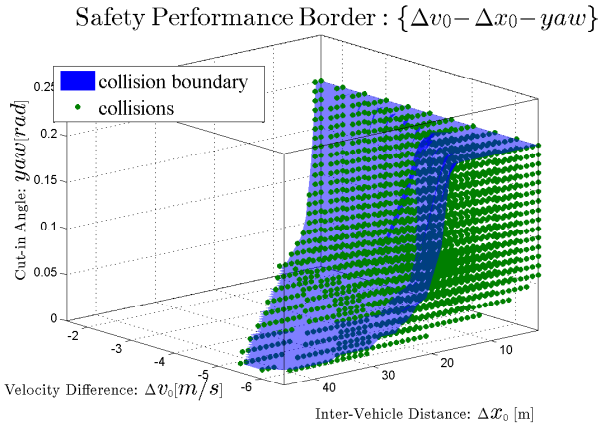

Fig. 1. The collision boundary for test case "cut in" resulting from grid searching

Against this background, we propose a criterion for evaluation of the approximation quality of the GPC based method in [8], based on the distribution of predictive probability generated by GPC within each iteration and the exit condition of the iteration. We apply this iterative algorithm on non-

[1]The authors are with the Johannes Kepler University, A-4040 Linz, Austria. Jinwei.Zhou@jku.at, Luigi.delre@jku.at,

This work has been partially conducted within the ENABLE-S3 project that has received funding from the Ecsel Joint Undertaking under grant agreement no 692455. This joint undertaking receives support from the European Union's Horizon 2020 Research and Innovation Programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, and Norway.

convex boundaries of both connected and unconnected sets, derived from the parameterization of a test case "cut in". The rest of this paper is organized as follows: section 2 introduces the modeling of the safety boundary. Section 3 clarifies GPC based Input Design strategies and the exit condition of the iteration. Then in section 4, we evaluate the performance of the algorithm and the exit condition on a test case example. Conclusions and discussions are given in section 5.

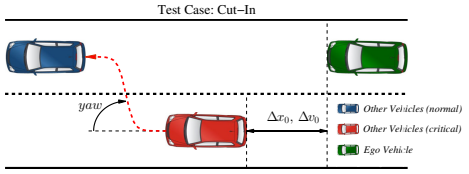

Fig. 2. The test case (cut-in): leading vehicle (blue), ego-vehicle (green), and cut-in vehicle (red)

II. Modeling of Safety Boundary

A. Preliminaries

For the safety assessment, we choose the collision as the unsafe condition so that safety evaluation becomes a binary classification problem. Fig. 3 shows the procedure of safety boundary searching, based on a simple parameterization of the test case "cut-in" with 3 dimensional inputs and binary outputs (collision or not), see Fig. 2.

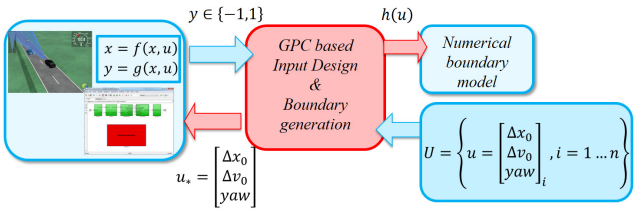

Fig. 3. State flow of safety boundary searching using GPC based Input Design

B. Safety boundary model

In this section, we explain the definition of the safety boundary. The parameterization of the test case "cut-in" is done by the following nonlinear model

$$\begin{aligned} \dot{\boldsymbol{x}} &= f(\boldsymbol{x}, \boldsymbol{u}) \\ y &= g(\boldsymbol{x}, \boldsymbol{u}), \end{aligned} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^{N_x}$, $\boldsymbol{u} \in \mathbb{R}^{N_u}$ and $y \in \{-1, 1\}$ are the state vector, input vector and output, respectively. The dimension N_u of the input vector depends on the parameterization of the test case. For example, in [6] the test case cut-in is parameterized through the longitudinal inter-vehicle distance Δx_0, the longitudinal velocity difference between ego vehicle and cut-in vehicle Δv_0 at the beginning of cut-in maneuver, and the cut-in angle yaw of the cut-in vehicle, namely $\boldsymbol{u} = [\Delta x_0, \Delta v_0, yaw]^T$. The output y is chosen as a binary variable, which represents the safety output: either safe $y = -1$ or collision $y = 1$, see Fig. 3. $\boldsymbol{x}$ can be any variables related to vehicle dynamics and interaction between vehicles, which are not of interest for us. Thus, the boundary can be defined as follows:

$$h(\boldsymbol{u}) \begin{cases} \leq 0 & \text{admissible region} \\ = 0 & \text{boundary of the admissible region,} \\ > 0 & \text{inadmmissible region} \end{cases} \tag{2}$$

where the decision vector of safety boundary is the input vector $\boldsymbol{u}$, see [8]. Since it is to apply the GPC based Input Design algorithm for boundary searching, we squash $h(\boldsymbol{u})$ through a logistic function into the interval $[0,\ 1]$,

$$\begin{aligned} h_p(\boldsymbol{u}) &= \frac{\eta}{\eta + exp(-h(\boldsymbol{u}))} \\ p_{th} &= h_p(\boldsymbol{u})|_{h(\boldsymbol{u})=0} = \frac{\eta}{\eta + 1},\ \eta > 0 \end{aligned} \tag{3}$$

where η is mandatory positive, so that it can be derived directly from the predictive probability p over U, the set of all values of the input. p_{th} is a probability threshold for the boundary, which determines the aggressiveness of algorithm during input space exploration.

$$h_p(\boldsymbol{u}) \begin{cases} \in [0,\ p_{th}] & \text{admissible region} \\ = p_{th} & \text{boundary of the admissible region} \\ \in (p_{th},\ 1] & \text{inadmmissible region} \end{cases} \tag{4}$$

The details will be illustrated in the next section.

III. Safety Boundary Searching

In order to characterize the border p_{th} we look at the boundary condition H_0 that lead to it, *i.e.* $H_0 = \{\boldsymbol{u} | h(\boldsymbol{u}) = 0\}$. However, this can be a very time consuming test, *e.g.* Fig. 1 shows the collision boundary for test case "cut in" resulting from the grid searching (the number of grid points > 10000) given in [6], where the shape and the size of the collision boundary were unknown in advance. This procedure is time consuming, even in a simulation environment. In order to reduce to time, we suggest to apply an Input Design strategy based on GPC.

In following we introduce a GPC based Input Design strategy for boundary searching of the connected set, [8] [9]. We propose an exit condition for the iteration algorithm, and apply the algorithm on a non-convex, unconnected set.

A. Binary Gaussian Process Classification

The classification problem is to output the correct class label for a new input data through the knowledge obtained from the existing data (training data set). The training data set can be any classified measurements, simulation results or knowledges. It can be expressed as $D = \{\bar{\boldsymbol{u}}, \bar{y}\}$, where $\bar{\boldsymbol{u}} \in \mathbb{R}^{N_u \times N_D}$, $\bar{y} \in \{-1, 1\}^{N_D}$ and N_D denote the input data, output class labels and the number of samples, respectively.

Gaussian Process Classification (GPC) is a nonparametric classification method based on a bayesian methodology. In GPC, it is assumed that the probability of a class label is

monotonically related to the value of some latent real value function $f_l(\boldsymbol{u}) \in \mathbb{R}$ at that location, namely

$$p(y=+1|\boldsymbol{u},D,f_l(\boldsymbol{u})) = p(y=+1|f_l(\boldsymbol{u})) \tag{5}$$

We assume a Gaussian Process (GP) prior over f_l and squash it through a logistic function to obtain the prior for binary output,

$$p(y=+1|f_l(\boldsymbol{u})) = \frac{1}{1+exp(-f_l(\boldsymbol{u}))} \tag{6}$$

In the following, we simplify the expression by using the notation $f_l = f_l(\boldsymbol{u})$, $\bar{f}_l = f_l(\bar{\boldsymbol{u}})$ and $f_{l*} = f_l(\boldsymbol{u}_*)$, where $\boldsymbol{u}_* \in U/\bar{\boldsymbol{u}}$ refers to new input data.

The inference is divided into 2 steps. First, we calculate the distribution of the latent variable corresponding to the new input data $\boldsymbol{u}_*$.

$$p(f_{l*}|\bar{\boldsymbol{u}},\bar{y},\boldsymbol{u}_*) = \int p(f_{l*}|\bar{\boldsymbol{u}},\bar{f}_l,\boldsymbol{u}_*)p(\bar{f}_l|\bar{\boldsymbol{u}},\bar{y})d\bar{f}_l \tag{7}$$

where $p(\bar{f}_l|\bar{\boldsymbol{u}},\bar{y}) = p(\bar{y}|\bar{f}_l)p(\bar{f}_l|\bar{\boldsymbol{u}})/p(\bar{y}|\bar{\boldsymbol{u}})$ is the posterior over the latent variables. $p(\bar{y}|\bar{f}_l)$, $p(\bar{f}_l|\bar{\boldsymbol{u}})$, and $p(\bar{y}|\bar{\boldsymbol{u}})$ are the likelihood function, the marginal likelihood and GP prior of f_l, respectively. Subsequently, we use the probability of f_{l*} to predict the probability of output class label for the corresponding new input data $\boldsymbol{u}_*$ as follows:

$$p(y_*=+1|\bar{\boldsymbol{u}},\bar{y},\boldsymbol{u}_*) = \int p(y_*|f_{l*})p(f_{l*}|\bar{\boldsymbol{u}},\bar{y},\boldsymbol{u}_*)df_{l*}\,, \tag{8}$$

As for the latter one, due to the non-gaussian likelihood in (7) and (8), the integrals are not analytically tractable. Therefore the analytical approximation methods are applied. Two common analytical approximation methods are the Laplace and the expectation propagation approximations, see [9]. In this work, we apply Laplace method, using a Gaussian Approximation $q(\bar{f}_l|\bar{\boldsymbol{u}},\bar{y})$ of the posterior $p(\bar{f}_l|\bar{\boldsymbol{u}},\bar{y})$ in Eq.(7). Accordingly, the prediction of the new input data $\boldsymbol{u}_*$ is evaluated through Gaussian Approximation:

$$p(y_*=+1|\bar{\boldsymbol{u}},\bar{y},\boldsymbol{u}_*) \approx \int p(y_*|f_{l*})q(f_{l*}|\bar{\boldsymbol{u}},\bar{y},\boldsymbol{u}_*)df_{l*}\,, \tag{9}$$

where $q(f_{l*}|\bar{\boldsymbol{u}},\bar{y},\boldsymbol{u}_*)$ is gaussian with $\mathcal{N}(E_q, V_q)$ given by:

$$\begin{aligned} E_q[f_{l*}|\bar{\boldsymbol{u}},\bar{y},\boldsymbol{u}_*] &= \bar{k}_*^T K^{-1}\hat{\bar{f}}_l \\ V_q[f_{l*}|\bar{\boldsymbol{u}},\bar{y},\boldsymbol{u}_*] &= k(\boldsymbol{u}_*,\boldsymbol{u}_*) - \bar{k}_*^T\left(K+W^{-1}\right)\bar{k}_*\,, \end{aligned} \tag{10}$$

in which $\hat{\bar{f}}_l = arg\,max_{\bar{f}_l}\, p\left(\bar{f}_l|\bar{\boldsymbol{u}},\bar{y}\right)$ and $W \triangleq \nabla^2 \log p\left(\bar{y}|\bar{f}_l\right)$. $\bar{k}_* = k(\boldsymbol{u}_*)$ and $K = K(\bar{\boldsymbol{u}},\bar{\boldsymbol{u}})$ denote the vector of covariances between the new input point and the N_D training points, and the covariance matrix of N_D training points, respectively.

Thus, the predictive probability p of an output belonging to a class is approximated through the averaged predictive probability given by Eq. (9) and Eq. (10).

B. Input Design strategy

In addition to the predictive probability model, which predicts the probability of an output belonging to a class for the corresponding input data, GPC generates an approximate predictive variance model as well, due to the Gaussian Approximation of the posterior, see (10). The idea is that inside

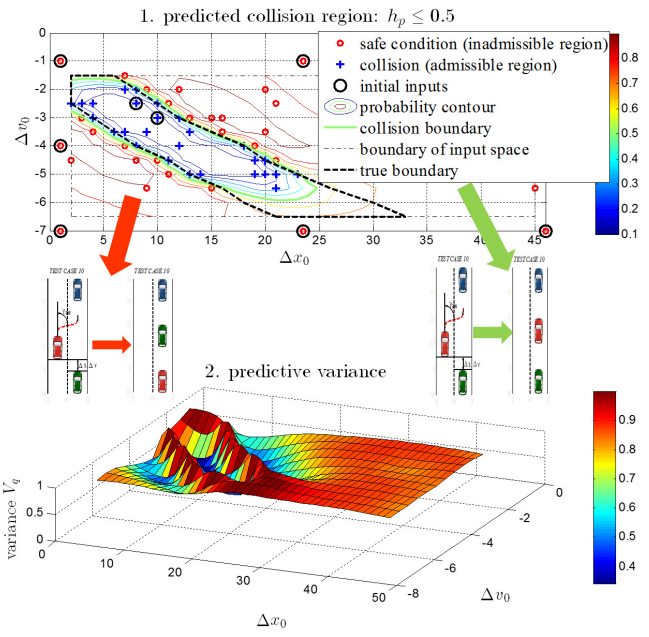

Fig. 4. The predictive collision boundary $p = 0.5$ and the corresponding predictive variance after 50 iteration

the admissible region $U_* = \{\boldsymbol{u}_*|p_* \leq p_{th})\}$, the predictive variance $V_q[f_{l*}|\bar{\boldsymbol{u}},\bar{y},\boldsymbol{u}_*]$ will have larger values on the class boundary. p_{th} is a probability threshold for the boundary. It measures how confident the predictive probability p_* is.

Using predictive variance, we can determine the new input point in the admissible region U_*. The boundary threshold determines how aggressive the searching is. A larger p_{th} speeds up the space exploration, however, it is more likely to locate the new input point in the inadmissible region, which is unwanted in some Hardware-in-the-Loop simulation due to the risk of hardware damage, *e.g.* engine test bed introduced in [8] [7].

Fig. 4 illustrates the output of GPC after 50 iterations based on the test case (cut-in) introduced in section 2. For visualization purpose, we fix the cut-in angle and reduce the input to a 2 dimensional vector, namely $\boldsymbol{u} = [\Delta x_0, \Delta v_0]$. The initial input points are collisions. The set of collisions is the admissible region, which is a connected set with non-convex boundary. Panel (1) shows the class boundary separating the safe conditions from collisions, which refers to 2 different types of the safe condition. The corresponding predictive variance is shown in panel (2). After the iterations stop, we obtain the admissible region $\bar{U} = \{\boldsymbol{u}|p \leq p_{th}\}$ and accordingly the boundary set $U_0 = \{\boldsymbol{u}|p = p_{th}\}$ is the approximation of the true class boundary $H_0 = \{\boldsymbol{u}|h(\boldsymbol{u}) = 0\}$.

The GPC based iterative algorithm for boundary searching can be summarized as follows:

1) set some classified initial points $\{\bar{\boldsymbol{u}},\bar{y}\}$ and some forbidden points with adverse class label outside the inputs boundary, see Fig. 4
2) generate the class boundary using GPC based on $\{\bar{\boldsymbol{u}},\bar{y}\}$

and find the admissible region $U_* = \{\boldsymbol{u}_* | p_* \leq p_{th})\}$

3) choosing the new input data according to $\boldsymbol{u}_* = \arg\max_{\boldsymbol{u}_* \in U_*} V_q$ and find the class label of y through simulation or experiment
4) stop iteration if the exit condition Eq. (12) is satisfied, otherwise start from 2) again
5) generate class boundary numerically

C. Approximation quality and the exit condition

Due to several reasons, it might be good to choose the exit condition as a predefined number of iterations n_{iter} as in [7], [8]. As a result the approximation quality is unknown. In our case it is desirable to find an exit condition depending on the current quality of the approximation.

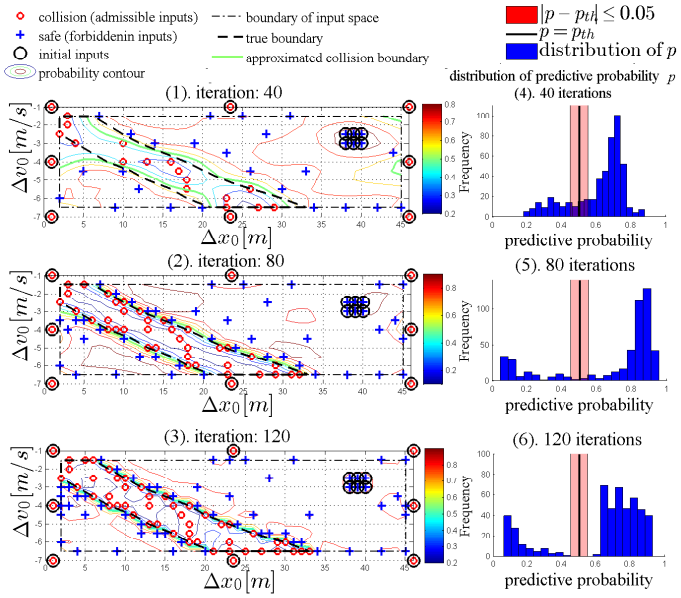

Fig. 5. The class boundary and the distribution of predictive probability after different iteration steps for $p_{th} = 0.5$.

In this work, we use the distribution of predictive probability as the indicator of the approximation quality, that decides when to exit the iteration. Fig. 5 shows the simulation results of the same test case, mentioned above. However, the initial inputs are the only safe conditions. The set of safe conditions is the inadmissible region. Comparing panel $(4)-(6)$ in Fig. 5, the predictive probability tends to migrate to either $p = 0$ or $p = 1$ over the iterations. After a certain number of iterations, they are separated by $p = p_{th}$ and the true boundary is well approximated by the class boundary, see Fig. 5 panel $(1)-(3)$. In Fig. 5 panel (3), we notice that most of the adjacent point pairs (input points next to each other) with adverse class labels are tested and the contour of the $p = p_{th}$ goes through all these adjacent point pairs. The input points are placed exactly on or next to the true boundary after 120 iterations Fig. 6 illustrates the local zoom of panel (3) in Fig. 5. The class boundary is very close to the true boundary. Thus, the predictive probability

$$p(y_* = +1 | \bar{\boldsymbol{u}}, \bar{y}, \boldsymbol{u}_*) \neq p_{th} \quad \forall \boldsymbol{u}_* \in U / \bar{\boldsymbol{u}} \tag{11}$$

represents the exit condition of the iteration. However, it is not applicable in practice because the numerical error and discretization of input space cause mis-triggering of this exit condition. Moreover, Eq. (11) is not applicable for approximation quality estimation.

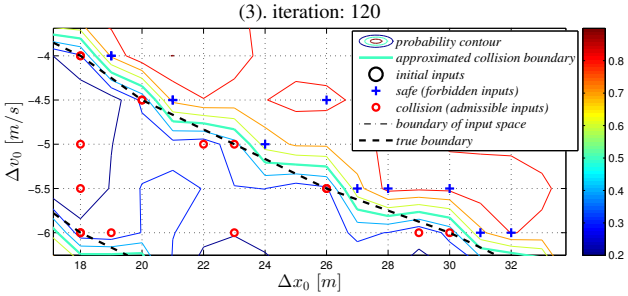

Fig. 6. A local zoom in of panel 3). in Fig. 5

To overcome this limitation, we now address a predictive probability margin $|p - p_{th}| \leq \Delta p$ instead of the separating contour $p = p_{th}$ as the indicator of approximation quality, so that it is robust against mis-triggering of the exit condition. As shown in Fig. 5 panel $(4)-(6)$ the predictive probabilities are distributed outside the region $p \in [p_{th} - 0.05,\ p_{th} + 0.05]$. Furthermore, we express it in percentage as follow:

$$U_{th\%} = \frac{|U_{th}|}{|U|} \times 100 \leq n_{th\%}, \qquad U_{th} = \{\boldsymbol{u} \mid |p - p_{th}| \leq \Delta p\} \tag{12}$$

where $|U|$ and $|U_{th}|$ are the cardinality of U and U_{th}, respectively . Fig. 7 shows the change of $U_{th\%}$ over iteration steps. With the decreasing value of $U_{th\%}$, the approximation quality increases. We find that $U_{th\%}$ approaches zero after 90 iterations and keeps zero after 116 iterations. The true boundary is approximated with high precision.

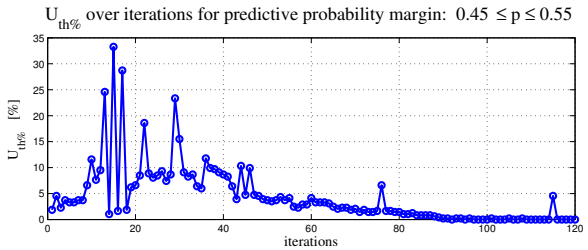

Fig. 7. percentage of the specified predictive probability interval over iteration steps

In practice, a non-zero positive threshold, *e.g.* $n_{th\%} = 2\,\%$, may reduce the iteration number enormously but still guarantee a good approximation of the true boundary, e.e. the approximated boundary in Fig. 5 panel (2) and (3) . In

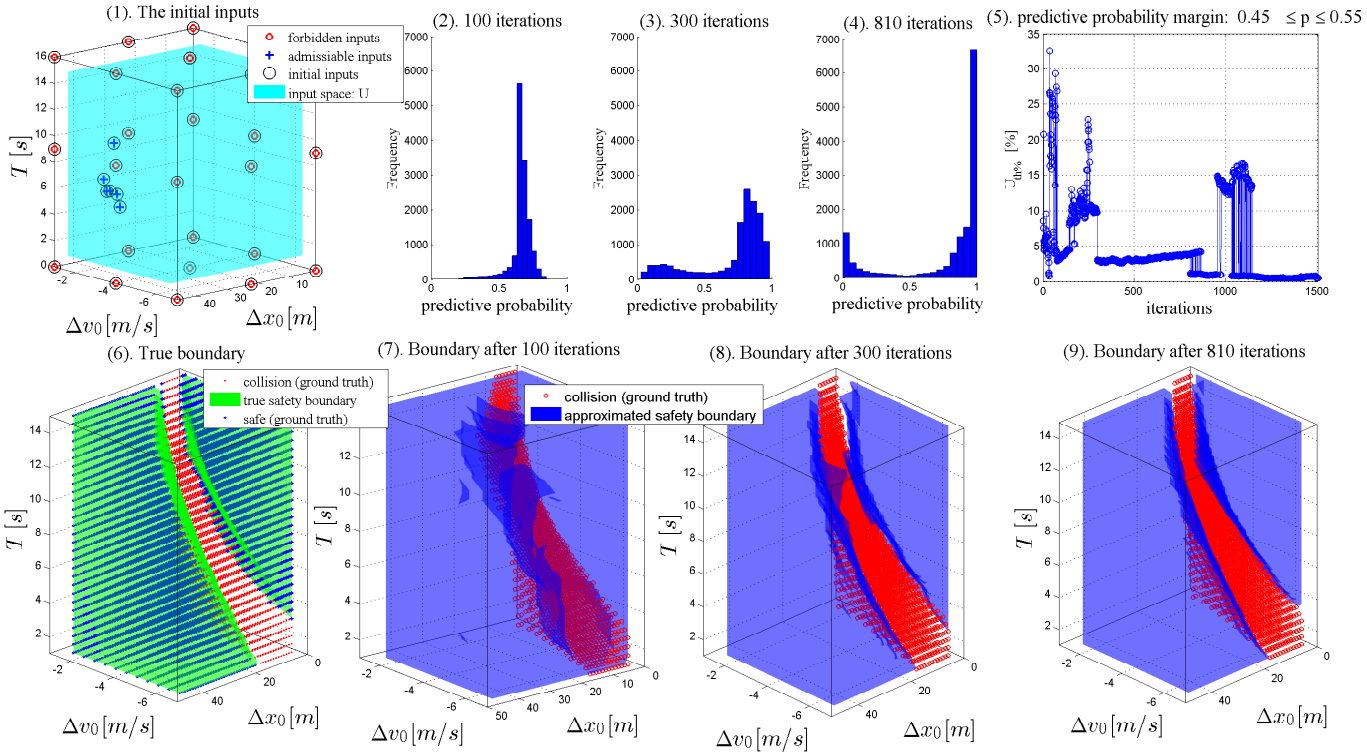

Fig. 8. Comparison of simulation results after different iteration steps

this case either a minimum number of iteration steps has to be set in advance *e.g.* 5 % $|U|$, or a filter to smooth the $U_{th\%}$ curve should be employed, so that the iteration would not be interrupted by mistake at the beginning, see Fig. 7.

In fact, the approaching can be further accelerated, if we force the algorithm to pick the new inputs within a given candidate set $U_{th\%}$, $U_{th} = \{\boldsymbol{u} \mid |p - p_{th}| \leq \Delta p\}$, if it is available. This will drive $U_{th\%}$ to zero as soon as possible. This approach can not only reduce the iteration steps but also improve the approximation quality in the case of approximation of a non-smooth boundary. This will be clarified in section 4 on a test case example.

In conclusion, the GPC based Input Design strategy can explore the design space to search for the boundary of both connected and unconnected set efficiently. Moreover, we propose an exit condition for the iteration algorithm, in which the distribution of predictive probability over the iteration steps is selected as the indicator of approximation quality.

IV. Example: Safety Boundary

In this section, we apply the GPC based Input Design method on the test case "cut in". We test the algorithm for the initial input points of safe conditions, namely the region of safe condition is the admissible region.

The test case is defined as follows: the ego-vehicle is following a leading vehicle in a suitable distance with $t_{headway} = 1.5s$, when suddenly a vehicle from the adjacent lane cuts in with a slower but constant velocity, see Fig. 2. Δx_0 and Δv_0 are inter-vehicle distance and velocity difference between ego-vehicle and cut-in vehicle at the beginning of lane change, respectively. The cut-in angle *yaw* is converted into equivalent lane change duration T, which characterize how aggressive the "cut in" maneuver is [10]. It is to mention that a non-aggressive, safe "cut in" maneuver can cause rear end collision with vehicle with ACC as well [6]. The trajectory of cut-in vehicle is given through a parameterized hyperbolic tangent function. Finally, a 3-dimensional input space $\{\Delta x_0, \Delta v_0, T\}$, derived from measurements [10], is defined as follows:

$$\begin{aligned} \Delta x_0 &\in [2,\ 45]\, m \\ \Delta v_0 &\in [-6.5,\ -1.5]\, m/s \\ T &\in [1,\ 15]\, s \end{aligned} \tag{13}$$

with the step size of $\{1m,\ 0.5m/s,\ 0.5s\}$. As a result, one has $44 \times 11 \times 29$ grid points. The true boundary is obtained through the grid searching of 14036 grid points over the whole input space U, namely $|U| = 14036$ see Fig. 8 panel (6) . The simulation runs on *Matlab* using *GPML Matlab* package [11] and *IPG CarMaker*. The ego-vehicle is equipped with the same ACC system as in [6]. The GPC based Input Design algorithm and the exit condition are applied. The minimum number of iteration steps is chosen as $n_{iter} \geq 5\ \%\ |U| \approx 700$ and the threshold of exit condition is chosen as $U_{th\%} \leq 2\%$ for the predictive probability threshold $p_{th} = 0.5$ with margin 0.05, namely $U_{th} = \{\boldsymbol{u} \mid |p - p_{th}| \leq 0.05\}$. 6 classified initial inputs are placed in the admissible region (safe condition, class label -1) and 26 forbidden points (class labels $y = +1$) are placed

outside U, see Fig. 8. The set of collisions is the inadmissible region. Furthermore it is to mention that in this case the admissible region is an unconnected set. It is separated by the inadmissible region (collision) into 2 parts.

As shown in Fig. 8 panel $(7)-(9)$ the class boundary approaches the true boundary quickly after 300 iterations. The new input points are placed near the true boundary. The iteration stops at $n_{iter} = 810$ due to the exit condition. We achieve a good approximation of the true boundary through the class boundary. The corresponding distributions of the predictive probability are shown in Fig. 8 panel $(2)-(4)$. Obviously, the predictive probability tends to shift to either $p = 1$ or $p = 0$ over the -iteration time. To clarify the proposed exit condition we ran the simulation for a fixed iteration number $n_{iter} = 1500$. As shown in Fig. 8 panel (5), $U_{th\%}$ tends to approach zero over the iteration steps. However, we notice that it rebounds to a high value at $n_{th} \in [950, 1140]$. This is due to the non-smoothness on the edge of the true boundary. When the algorithm tries to approach these edges of the true boundary, the approximation quality may get worse temporarily. The approximation quality will recover in a few iterations, see Fig. 9.

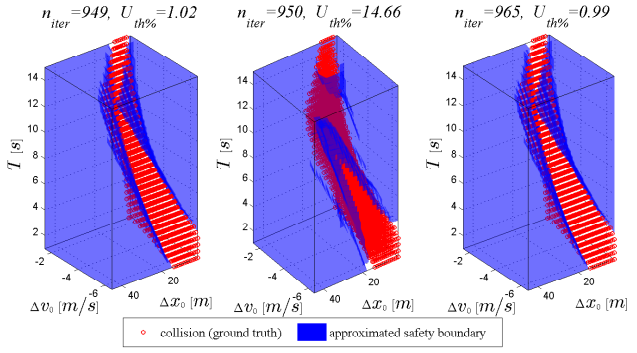

Fig. 9. Comparison of class boundary and $U_{th\%}$ for $n_{iter} \in \{949, 950, 965\}$

As mentioned in section 4, the candidate set U_{th} helps to accelerate the approaching and improve the quality of approximation. The $U_{th\%}$ of the algorithm with candidate set U_{th} is compared with the original algorithm in Fig. 10. We find from panel (1), that with candidate set U_{th}, $U_{th\%}$ has higher values at the beginning of iterations and approaches zero more quickly. The rebound of $U_{th\%}$ appears earlier and the duration is much shorter than the original algorithm. Furthermore, we can find from panel (2), a local zoom near zero line, that $U_{th\%}$ of the algorithm with candidate set U_{th} approaches zero after 1150 iterations, while $U_{th\%}$ of the algorithm without candidate set U_{th} does not approaches zero within 1500 iterations (minimum of $U_{th\%}$ is 1.17).

V. CONCLUSIONS

In this work, we applied the iterative GPC based Input Design method on the safety boundary searching problem. We proposed a criteria for the evaluation of the approximation quality of the class boundary based on the distribution of the predictive probability generated by GPC. According to

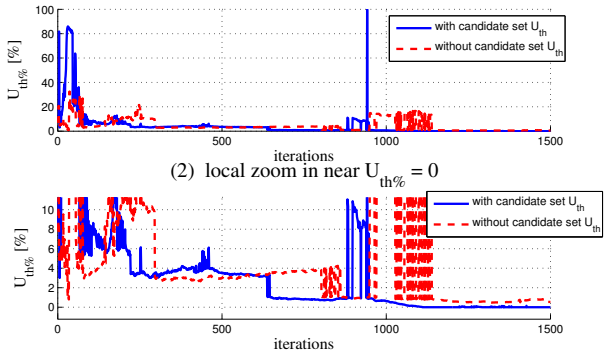

Fig. 10. Comparison of $U_{th\%}$ based on iteration algorithm with and without candidate set U_{th}

the approximation quality, an exit condition for the iterative algorithm was proposed, so that the iteration stops automatically when the given approximation precision is achieved. Besides we employed a candidate set U_{th} set to speed up the zero approaching of the $U_{th\%}$, which guarantees the precise approximation of true boundary. The proposed method is also suitable for higher input dimensions.

The future work will focus on the application of the proposed method for more complicated test cases for safety testing of ADAS/ADFs.

REFERENCES

[1] H. Winner, "Safe driving despite uncertainties," in *ASAM International Conference*, Dresden, 2013. [Online]. Available: http://tubiblio.ulb.tu-darmstadt.de/63813/

[2] O. Schädler, S. Müller, and M. Gründl, "Experimental evaluation of the controllability of interacting advanced driver assistance systems," in *The Dynamics of Vehicles on Roads and Tracks: Proceedings of the 24th Symposium of the International Association for Vehicle System Dynamics (IAVSD 2015), Graz, Austria, 17-21 August 2015*. CRC Press, 2016, p. 297.

[3] F. Schuldt, F. Saust, B. Lichte, M. Maurer, and S. Scholz, "Effiziente systematische testgenerierung für fahrerassistenzsysteme in virtuellen umgebungen," *Automatisierungssysteme, Assistenzsysteme und Eingebettete Systeme Für Transportmittel*, 2013.

[4] Y. Shoukry, P. Tabuada, S. Tsuei, M. B. Milam, J. W. Grizzle, and A. D. Ames, "Closed-form controlled invariant sets for pedestrian avoidance," in *2017 American Control Conference (ACC)*, May 2017, pp. 1622–1628.

[5] D. W. Oyler, Y. Yildiz, A. R. Girard, N. I. Li, and I. V. Kolmanovsky, "A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 1705–1710.

[6] J. Zhou and L. del Re, "Reduced complexity safety testing for adas & adf," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5985–5990, 2017.

[7] G. Kampmann, N. Kieft, and O. Nelles, "Support vector machines for design space exploration," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 2, 2012, pp. 1116–1121.

[8] H. Oyama, M. Yamakita, K. Sata, and A. Ohata, "Identification of static boundary model based on gaussian process classification," *IFAC-PapersOnLine*, vol. 49, no. 11, pp. 787–792, 2016.

[9] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[10] J. Zhou and L. del Re, "Identification of critical cases of adas safety by fot based parameterization of a catalogue," in *2017 11th Asian Control Conference (ASCC)*, Dec 2017, pp. 453–458.

[11] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning software*. [Online]. Available: http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html