TRELPy: Toolbox for Task-Relevant Evaluation of Perception

Ranai Srivastav[1], Apurva Badithela[2], Tichakorn Wongpiromsarn[1], Richard M. Murray[2]

***Abstract*—In safety-critical autonomous systems, deriving system-level guarantees requires evaluation of individual subsystems in a manner consistent with the system-level task. These safety guarantees require careful reasoning about how to evaluate each subsystem, and the evaluations have to be consistent with subsystem interactions and any assumptions made therein. A common example is the interaction between perception and planning. `TRELPy` is a Python-based toolbox that can evaluate the performance of perception models and leverage these evaluations in computing system-level guarantees via probabilistic model checking. The tool implements this framework for popular detection metrics such as confusion matrices, and implements new metrics such as proposition-labeled confusion matrices. The propositional formulae for the labels of the confusion matrix are chosen such that the confusion matrices are relevant to the downstream planner and system-level task. `TRELPy` can also group objects by egocentric distance or by orientation relative to the ego vehicle to further make the confusion matrix more task relevant. These metrics are leveraged to compute the combined performance of the perception and planner and calculate the satisfaction probability of system-level requirements.**

I. Introduction

`TRELPy`(https://tinyurl.com/TRELPy) is a toolbox that provides methods to compute metrics that evaluate the performance of learning-based perception models (specifically the detection task of perception) and use these metrics in computing system-level guarantees via probabilistic model checking. The tool allows for defining new evaluation metrics that are better suited for the task and abstraction levels corresponding to other subsystems. Traditional metrics such as confusion matrices (CMs) with object class labels are conservative in their evaluation of performance of detection models because they assign equal importance to each detection and misdetection. `TRELPy` overcomes this by defining the following metrics that are task relevant i) CM with propositional formulae as labels ii) Class-labeled CM, along with options to group evaluations by a) distance from ego b) orientation with respect to ego.

For example, distance-based grouping of evaluations is relevant for planning tasks where accurately detecting nearby objects is important. Conversely, correct decision-making by a high-level planner might not need accurate detections of every object. By leveraging the confusion matrix as a probabilistic model of sensor error, we construct a Markov chain model of system-state evolution to capture the impact of perception errors on planning. Then, we use an off-the-shelf probabilistic model-checker, STORM [1], to compute the system-level probability of satisfying the task with the given perception model and a discrete-state planner. The planner can either be synthesized correct-by-construction for the system-level task or user-defined. The computed probability thus evaluates a system-level guarantee by accounting for perception performance and its effect on planner outcomes.

II. Demonstration Plan

We plan to present the pipeline presented in Figure 1.

Example: We define an environment with a car moving down a straight road, approaching a crosswalk. The car safety specification states the car must continue driving unless a pedestrian is on the crosswalk, in which case, it must stop right before the crosswalk. The controller is synthesized

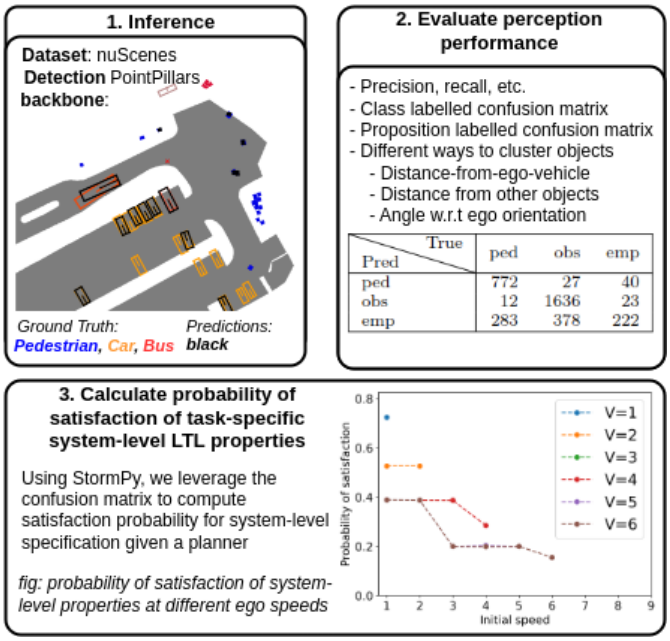

Fig. 1: Overview of TRELPy

to meet these requirements. The confusion matrices are generated using the validation split of the NuScenes Dataset [2] using the pretrained PointPillars [3] LIDAR model found in MMDetection3D [4].

During the demonstration, we plan to go over steps to setup the repository to run inference and configure necessary thresholds and output directories. The user can also configure distance thresholds for evaluation, the propositional formulas, and the mapping from the detection model's outputs to these propositional formulas, and choose to use a synthesized controller or a custom controller provided by the package. Finally, using `STORM` [1] and `TuLiP` [5], `TRELPy` computes the task-relevant safety guarantee via probabilistic model checking.

References

[1] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A STORM is coming: A modern probabilistic model checker," in *Computer Aided Verification* (R. Majumdar and V. Kunčak, eds.), (Cham), pp. 592–600, Springer International Publishing, 2017.

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[3] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12697–12705, 2019.

[4] M. Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection." https://github.com/open-mmlab/mmdetection3d, 2020.

[5] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLiP: a software toolbox for receding horizon temporal logic planning," in *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, HSCC '11, p. 313–314, Association for Computing Machinery, 2011.

[1] Department of Computer Science, Iowa State University, Ames, IA. Acknowledging funding from grant NSF CNS-2141153 ({ranais, nok}@iastate.edu)

[2] Controls and Dynamical Systems, California Institute of Technology, Pasadena, CA. Acknowledging funding from AFOSR FA9550-22-1-0333 - ({apurva, murray}@caltech.edu)