Evaluation Metrics of Object Detection for Quantitative System-Level Analysis of Safety-Critical Autonomous Systems

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

***Abstract*— This paper proposes two new metrics to evaluate learned object detection models that can be leveraged for quantitative system-level analysis via probabilistic model-checking. We define *proposition-labeled* and *distance-parametrized* confusion matrices and show that these matrices can be used to compute the probability of the closed-loop system satisfying its system-level specifications expressed in temporal logic. Instead of using object class labels, the proposition-labeled confusion matrix uses atomic propositions relevant to the high-level planning strategy. Furthermore, unlike the traditional confusion matrix, the proposed distance-parametrized confusion matrix accounts for variations in detection performance with respect to the distance between the ego and the object. We empirically show that these evaluation metrics chosen with the context of i) system-level specifications and ii) the planning module lead to a less conservative analysis in comparison to canonical metrics that do not take these into account. We demonstrate this framework on a car-pedestrian example by computing the satisfaction probabilities for safety requirements formalized in Linear Temporal Logic (LTL).**

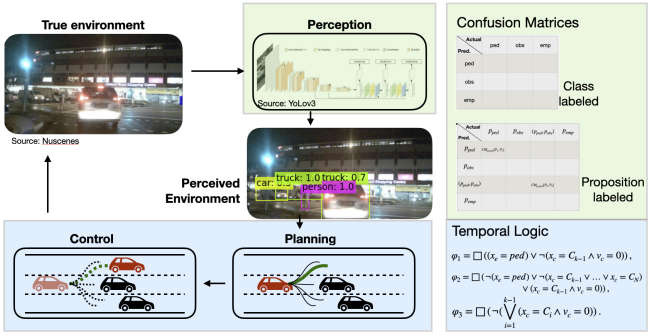


Fig. 1: Planning and control can be evaluated using temporal logic. However, this analysis often excludes perception performance. Based on confusion matrices from computer vision, we propose new metrics for object detection, and outline a framework for accounting for perception performance in evaluating the overall system.

I. Introduction

Formal verification is an important technique to ensure the correctness of safety-critical autonomous systems. However, when verifying learning-based components such as perception, two challenges arise: i) correctly specifying the formal requirements to be verified, and ii) verification of the component. In this paper, we evaluate learning-based object detection and classification models with respect to system-level safety requirements encoded in Linear Temporal Logic (LTL). Inspired by the use of confusion matrices in machine learning and computer vision [1]–[4], we define two new evaluation metrics for object detection. We provide a framework for coupling these metrics with planning and control to provide a quantitative metric of safety at the overall system-level.

In a typical autonomy software stack, the perception component is responsible for parsing sensor inputs to perform tasks such as object detection and classification, localization, and tracking, and the planning module is responsible for mission planning, behavior planning, and motion planning [5]. Further downstream in the autonomy stack, we have controllers to actuate the system to execute plans consistent with the planning module. For high-level, discrete decision-making tasks in robotics, using formal methods to specify requirements, synthesize controllers, and verify system models has been an effective paradigm [6]–[10]. Some of this success can be attributed to the ease of specifying safety, progress, and fairness properties at the system-level [11]–[16]. For instance, it is feasible to formally specify "*maintain safe longitudinal distance*" as a safety property for a self-driving car [14], [16]. However, using the same paradigm to evaluate perception can be challenging since it is difficult to formally characterize "correct" performance (relative to human-level perception) for perception tasks such as object detection [17]. Consider the task of classifying handwritten digits in the MNIST dataset [18] — it is not practical to formally specify the pixel configuration to correctly classify a digit. Similarly, for urban driving scenarios, it becomes impractical to specify formal requirements for object detection, and subsequently verify learned detection models. In computer vision, object detection models are evaluated using metrics such as accuracy, precision and recall [1], [19]. Several of these metrics are derived from the confusion matrix, which is obtained from the detection model's performance on an evaluation set [1], [19]. The work in [20], [21] shows that confusion matrices can be leveraged to account for object detection performance in a quantitative evaluation of the overall system with respect to system-level formal requirements.

However, confusion matrices, as traditionally studied in computer vision, are often constructed assuming detection performance on all datapoints is of equal importance. In a system-level analysis, this would not necessarily highlight detection errors that are safety-critical. For example, consider a cluster of pedestrians waiting at a crosswalk and an autonomous car with the requirement to come to a safe

We acknowledge funding from AFOSR Test and Evaluation Program, grant FA9550-19-1-0302, and from NSF grant CNS-2141153.

[1] Control and Dynamical Systems, California Institute of Technology.
[2] Department of Computer Science, Iowa State University.
*Corresponding author: A. Badithela apurva@caltech.edu

stop. If detecting one pedestrian vs. a group of pedestrians results in the same planning outcome of the car coming to a stop, then accounting for every missed pedestrian in the quantitative analysis would not accurately reflect the actual probability that the car behaves safely. Therefore, we argue that metrics for evaluating object detection models must be carefully chosen depending on the downstream planning logic and the system-level safety specifications. Note that this still allows for the design and evaluation of the detection module to be independent from the rest of the autonomy stack — only the choice of metrics is informed by the system-level specification and the decision-making module.

Given this context, our contributions are the following. We focus on the object detection task of perception, and use it to refer to both the tasks of detecting an object and correctly classifying it. First, we propose a *distance-parametrized* variation of the traditional confusion matrix to account for the effect of distance on object detection performance. Second, we define a *proposition-labeled* confusion matrix. Third, we use the proposed metrics to evaluate a pre-trained YoLov3 model [22] on the NuScenes [23] dataset. The probability that the overall system satisfies its specifications is then computed using probabilistic model checking via methods introduced in prior work [20], [21], [24]. Finally, we provide an empirical comparison of the quantitative analysis resulting from the different metrics for object detection on a car-pedestrian example. We show that the detection metrics that are *distance-parametrized* and *proposition-labeled* lead to less conservative outcomes in the system-level evaluation.

II. Related Work

Evaluating and monitoring perception for safety-critical errors is an emerging research topic [14], [25], [26]. Perception is a complex subsystem responsible for tasks such as detection, localization, segmentation. These recent works have focused on evaluating object detection in the context of system-level safety. We follow this early work and focus on object detection task of perception, which refers to both detecting an object and classifying it correctly. As an initial stage of this study, we assume a static environment and perfect object localization. These assumptions can potentially be relaxed based on an analysis that takes into account partial observability of the environment [27], as discussed in Section VI.

The use of Markov chains for probabilistic reasoning about the correctness of high-level robot behaviors in the presence of perception errors was studied in [24]. However, the algorithms in [24] assumed knowledge of the probabilistic sensor model. Rigorously constructing these sensor models from confusion matrices was presented in [20]. In [21], this approach was further extended by providing confidence intervals on the probabilistic sensor models and was applied to a case study on guiding aircraft on taxiways introduced by Boeing [28].

For runtime monitoring of perception systems, Timed Quality Temporal Logic (TQTL) is used to specify spatio-temporal requirements on perception [29], [30] . However, to specify these requirements, the user has to label each scenario with critical objects that need to be detected. This approach is useful in evaluating perception in isolation with respect to the requirements defined on a specific scenario. In [26], temporal diagnostic graphs are proposed to identify failures in object detection during runtime.

In [25], Hamilton-Jacobi reachability was used to account for closed-loop interactions with agents in the environment to identify safety-critical perception zones in which correct detection is crucial. Our work can be viewed as a complementary approach to [25] by allowing crucial misclassifications, according to system-level analysis, to be identified.

III. Preliminaries

First, we give an overview of temporal logic, which is used to specify system-level requirements formally. Second, we describe the performance metrics used to evaluate object detection and classification models in the computer vision community. Finally, we setup a simple discrete-state car-pedestrian system as a running example to illustrate the role of these different concepts. Our approach can be applied to more complex systems, including those with continuous state, by applying state-space discretization as later explained in Remark 2.

A. Temporal Logic for Specifying System-level Properties

System Specification. We use the term system to refer to refer to the autonomous agent and its environment. The agent is defined by variables V_A, and the environment is defined by variables V_E. The valuation of V_A is the set of states of the agent S_A, and the valuation of V_E is the set of states of the environment S_E. Thus, the states of the overall system is the set $S := S_A \times S_E$. Let AP be a finite set of atomic propositions over the variables V_A and V_E. An atomic proposition $a \in AP$ is a statement that can be evaluated to *true* or *false* over states in S.

We specify formal requirements on the system in LTL (see [11] for more details). An LTL formula is defined by (a) a set of atomic propositions, (b) logical operators such as: negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), and implication ($\implies$), and (c) temporal operators such as: next ($\bigcirc$), eventually ($\diamond$), always ($\Box$), and until ($\mathcal{U}$). The syntax of LTL is defined inductively as follows: (a) An atomic proposition p is an LTL formula, and (b) if φ and ψ are LTL formulae, then $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$, $\varphi \,\mathcal{U}\, \psi$ are also LTL formulae. Further operators can be defined by a temporal or logical combination of formulas with the aforementioned operators. For an infinite trace $\sigma = s_0 s_1 \ldots$, where $s_i \in 2^{AP}$, and an LTL formula φ defined over AP, we use $\sigma \models \varphi$ to denote that σ satisfies φ. For example, the formula $\varphi = \Box p$ represents that the atomic proposition $p \in AP$ is satisfied at every state in the trace, i.e., $\sigma \models \varphi$ if and only if $p \in s_t, \forall t$. In this work, these traces σ are executions of the system, which we model using a Markov chain.

Definition 1 (Markov Chain [11])**.** A discrete-time Markov chain is a tuple $\mathcal{M} = (S, Pr, \iota_{init}, AP, L)$, where S is a non-empty, countable set of states, $Pr : S \times S \to [0, 1]$

is the *transition probability function* such that for all states $s \in S$, $\Sigma_{s' \in S} Pr(s, s') = 1$, $\iota_{init} : S \to [0, 1]$ is the initial distribution such that $\Sigma_{s \in S} \iota_{init}(s) = 1$, AP is a set of atomic propositions, and $L : S \to 2^{AP}$ is a labeling function. The labeling function returns the set of atomic propositions that evaluate to true at a given state. Given an LTL formula φ (defined over AP) that specifies requirements of a system modeled by the Markov Chain $\mathcal{M}$, the probability that a trace of the system starting from $s_0 \in S$ will satisfy φ is denoted by $\mathbb{P}_{\mathcal{M}}(s_0 \models \varphi)$. The definition of this probability function is detailed in [11].

B. Performance Metrics for Object Detection

We consider object detection to include both the detection and the classification tasks. In this section, we provide background on metrics used to evaluate performance with respect to these perception tasks. Let the evaluation dataset $\mathcal{D} = \{(f_i, b_i, d_i, x_i)\}_{i=1}^{N}$ consist of N objects across m image frames $F = \{F_1, \ldots, F_m\}$. For each object, $f_i \in F$ represents the image frame token, b_i specifies the bounding box coordinates, d_i denotes the distance of the object to ego, and x_i denotes the true class of the object. When a specific object detection algorithm is evaluated on $\mathcal{D}$, each object has a predicted bounding box, $\tilde{b}_i$, and predicted object class $\tilde{x}_i$. We store these predictions in the set $\mathcal{E} = \{(\tilde{b}_i, \tilde{x}_i)\}_{i=1}^{N}$.

Definition 2 (Confusion Matrix)**.** Let $\mathcal{D}$ be an evaluation set of objects and $\mathcal{E}$ be the corresponding predictions by an object detection algorithm. Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be a set of object classes in $\mathcal{D}$, and let n denote the cardinality of $\mathcal{C}$. The confusion matrix corresponding to the classes $\mathcal{C}$ and dataset $\mathcal{D}$, and predictions $\mathcal{E}$ is an $n \times n$ matrix $CM(\mathcal{C}, \mathcal{E}, \mathcal{D})$ with the following properties:

- $CM(\mathcal{C}, \mathcal{E}, \mathcal{D})[i, j]$ is the element in row i and column j of $CM(\mathcal{C}, \mathcal{E}, \mathcal{D})$, and represents the number of objects that are predicted to have class label $c_i \in \mathcal{C}$, but have the true class label $c_j \in \mathcal{C}$, and
- the sum of the j^{th}-column of $CM(\mathcal{C}, \mathcal{E}, \mathcal{D})$ is the total number of objects in $\mathcal{D}$ belonging to the class $c_j \in \mathcal{C}$.

Several performance metrics for object detection and classification such as true positive rate, false positive rate, precision, accuracy, and recall can be derived from the confusion matrix [1], [2], [19].

Remark 1. In this work, we use $c_n = empty$ (also abbreviated to *emp* in figures) as an auxiliary class label in the construction of confusion matrices. If an object has the true class label c_i but is not detected by the object detection algorithm, then this gets counted in $CM(\mathcal{C}, \mathcal{E}, \mathcal{D})[n, i]$ as a false negative with respect to class c_i. If the object was not labeled originally, but is detected and classified to have class label c_i, then it gets counted in $CM(\mathcal{C}, \mathcal{E}, \mathcal{D})[i, n]$ as a false negative of the *empty* class. We expect that in a properly annotated dataset, false negatives $CM(\mathcal{C}, \mathcal{E}, \mathcal{D})[i, n]$ to be small. We ignore these extra detections in constructing the confusion matrix because by not being annotated, they are not relevant to the evaluation of object detection models.

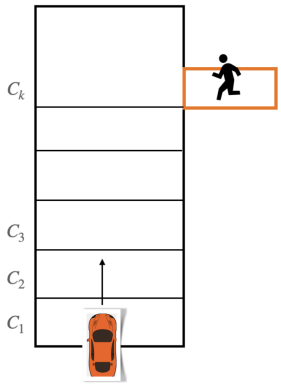


Fig. 2: Running example of a car and pedestrian. If there is a pedestrian at crosswalk cell C_k, that is, $x_e \models ped$, then the car must stop at cell C_{k-1}. Otherwise, it must not stop.

C. Example

Consider a car-pedestrian example, modeled using discrete transition system as illustrated in Figure 2. The true state of the environment is denoted by x_e, the state of the car comprises of its position and speed (x_c, v_c). The safety requirement on the car is that it "shall stop at the crosswalk if there is a waiting pedestrian, and not come to a stop, otherwise". The overall system specifications are formally expressed as safety specifications in equations 1-3. A more detailed description of this example can be found in [20].

(S1) If the true state of the environment is not a pedestrian, i.e. $x_e \neq ped$, then the car must not stop at C_{k-1}.

$$\varphi_1 = \square((x_e \neq ped) \to \neg(x_c = C_{k-1} \wedge v_c = 0)), \quad (1)$$

(S2) If $x_e = ped$, the car must stop on C_{k-1}.

$$\varphi_2 = \square\Big(x_e = ped \to ((x_c = C_{k-1} \wedge v_c = 0) \vee \neg(x_c = C_{k-1}))\Big), \quad (2)$$

(S3) The agent should not stop at any cell C_i, for all $i \in \{1, \ldots, k-2\}$,

$$\varphi_3 = \square\neg(\bigvee_{i=1}^{k-2}(x_c = C_i \wedge v_c = 0)). \quad (3)$$

The overall safety specification for the car is $\varphi := \varphi_1 \wedge \varphi_2 \wedge \varphi_3$. Since the car controller has been designed assuming perfect perception, the specification for the pedestrian and non-pedestrian environment simplifies to,

$$\varphi_{ped} = \square\neg(\bigvee_{i=1}^{k-2}(x_c = C_i \wedge v_c = 0)) \bigwedge \square(\neg(x_c = C_{k-1}) \vee (x_c = C_{k-1} \wedge v_c = 0)),$$

$$\varphi_{class} = \square\neg(\bigvee_{i=1}^{k-1}(x_c = C_i \wedge v_c = 0)),\ class \in \{obs, emp\}.$$

As mentioned previously, we assume a static environment. We also assume that the car knows the location of the crosswalk, e.g. from HD map information, and that it can coarsely

localize whether the detected object is on the crosswalk. The evaluation framework presented in this paper is valid for any discrete-state control strategy, both deterministic and probabilistic. To concretize the setup, we consider a car controller that acts corresponding to the detection model's prediction of the environment at the crosswalk. If the car at time step t detects a pedestrian, then it chooses its speed according to a control strategy for φ_{ped} to come to a stop before the crosswalk at cell C_{k-1}. If the state of the car is such that it is impossible to find a controller that will bring it to a stop at cell C_{k-1}, then it decelerates as fast as possible. Similarly, if an obstacle or empty sidewalk is detected, then the car chooses its speed according to a control strategy designed correct-by-construction for φ_k.

Remark 2. This paper proposes evaluation metrics for object detection in safety-critical autonomous systems to provide a probabilistic guarantee of the correctness of the overall closed-loop system with respect to its system-level specifications. These systems typically comprise of both continuous dynamics for low-level control of the physical system and discrete logic responsible for high-level decision-making. A common approach to integrate the reasoning of discrete and continuous behaviors is to construct a finite state model that serves as an abstract model of the physical system (which typically has infinitely many states), and formally verify the resulting finite state abstraction [31]–[38]. However, in this work, we focus on the high-level logic and we assume that the system has been abstracted using a finite state model.

IV. Metrics for Evaluating Object Detection

In this section, we present the construction of proposition-labeled and distance parameterized confusion matrices. The distance parametrization can be augmented to both the proposition-labeled confusion matrix and the more traditional class-labeled confusion matrix, as outlined in Algorithms 1 and 2, respectively. While the confusion matrix provides useful metrics for evaluating object detection models, we would like to use these metrics in evaluating the performance of the system with respect to formal constraints in temporal logic. In [20], an algorithm was provided for system-level analysis by accounting for classification performance using the canonical confusion matrix. For each confusion matrix, we evaluate the system using the framework introduced in [20], and compare the results.

A. Proposition-labeled Confusion Matrix

In many instances, the planner need not require correct detections of every object to find a high-level strategy that is consistent with system-level specifications. For instance, for the planner to decide to stop for a group of pedestrians 20m away, the object detection does not need to correctly detect each and every pedestrian. In terms of quantifying system-level satisfaction of safety requirements, it is sufficient for the object detection to identify that there *are* pedestrians 20m away, and not necessarily to correctly detect the precise number of pedestrians. Thus, we introduce the notion of using atomic propositions as class labels in the confusion matrix instead of the object classes themselves.

Let p_i be the atomic proposition: "there exists an object of class $c_i \in \mathcal{C}$", and let $\mathcal{P} = \{p_1, \ldots, p_n\}$ denote the set of all atomic propositions. Let $D_0 < D_1 < \ldots < D_k < \ldots < D_{k_{\max}}$ denote progressively increasing distances from the autonomous vehicle. Let $\mathcal{D}_k \subset \mathcal{D}$ be the subset of the dataset that includes objects that are in the distance interval $z_k = (D_{k-1}, D_k)$ from the autonomous system. Let $\mathcal{E}_k$ denote the predictions of the object detection algorithm corresponding to dataset $\mathcal{D}_k$. For each parameter k, we define the proposition-labeled confusion matrix $CM_{\text{prop},k} = CM_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}_k, \mathcal{D}_k)$ where the classes are characterized by the powerset of atomic propositions $2^{\mathcal{P}}$. Algorithm 1 shows the construction of the proposition-labeled confusion matrix.

Remark 3. In general, the set of atomic propositions $\mathcal{P}$ depend on the logic used by the planner to trigger different operation modes. In the running example (Section III-C), the planner outputs different actions depending on the environment, i.e, pedestrian or other objects. If the planner responds differently to other types of objects, e.g cars, bicycles, cones, those should be included in the set of atomic propositions $\mathcal{P}$. Thus, our approach can generalize to a wider range of scenarios by adapting the set $\mathcal{P}$ accordingly.

Algorithm 1 Proposition-labeled Confusion Matrix

procedure PropCM(Dataset $\mathcal{D} = \{(f_i, b_i, d_i, x_i)\}_{i=1}^{N}$, Classes $\mathcal{C}$, Distance Parameters $\{D_k\}_{k=0}^{k_{\max}}$)
  From $\{D_k\}_{k=0}^{k_{\max}}$, define distance intervals $\{z_k\}_{k=1}^{k_{\max}}$
  Run object detection algorithm to get predictions $\mathcal{E}$,
  Initialize $\mathcal{D}_1, \ldots, \mathcal{D}_{k_{\max}}$ as empty sets
  Initialize $\mathcal{E}_1, \ldots, \mathcal{E}_{k_{\max}}$ as empty sets
  for $(f_i, b_i, d_i, x_i) \in \mathcal{D}$ **do**
    if $d_i \in z_k$ **then**
      $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{(f_i, b_i, d_i, x_i)\}$
      $\mathcal{E}_k \leftarrow \mathcal{E}_k \cup \{(b_i, \tilde{x}_i)\}$
  for $c_j \in \mathcal{C}$ **do**
    $p_j \equiv$ "there exists an object of class c_j"
  $\mathcal{P} \leftarrow \bigcup_j \{p_j\}$ ▷ Set of atomic propositions
  for $k \in \{1, \ldots, k_{\max}\}$ **do**
    Denote $CM_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}_k, \mathcal{D}_k)$ as $CM_{\text{prop},k}$
    $CM_{\text{prop},k} \leftarrow$ zero matrix
    for $f \in F$ **do** ▷ Loop over image frames
      Group objects in $\mathcal{D}_k$ with image token f.
      Group predictions in $\mathcal{E}_k$ with image token f.
      $P_i \leftarrow$ Predicted set of propositions
      $P_j \leftarrow$ True set of propositions
      $CM_{\text{prop},k}[P_i, P_j] \leftarrow CM_{\text{prop},k}[P_i, P_j] + 1$
  $CM_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}, \mathcal{D}) = \{CM_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}_k, \mathcal{D}_k)\}_{k=0}^{k_{\max}}$
  return $CM_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}, \mathcal{D})$

The true environment is associated with a set of atomic propositions evaluating to true. Suppose, there is a pedestrian and a trash can in the distance interval z_k from the ego,

then the true class label is $\{p_{\text{ped}}, p_{\text{obs}}\}$ in the distance-parametrized confusion matrix $CM_{\text{prop},k}$. Note that for every possible environment, there is only one corresponding class in the proposition-labeled confusion matrix. Thus, for a given true environment, the predicted class of the environment at distance interval z_k could be any element of the set $2^{\mathcal{P}}$. Therefore, at each time step, the set of detection outcomes is $Outc = 2^{\mathcal{P}}$. The tuple $(Outc, 2^{Outc})$ forms a σ-algebra for defining a probability function over the proposition-labeled confusion matrix. Since the set $Outc$ is countable, we can define a probability function $\mu : Outc \to [0,1]$ such that $\sum_{e \in Outc} \mu(e) = 1$. For a distance-parametrized confusion matrix $CM_{\text{prop},k}$ with class labels in the set $Outc$, and for every true environment class label P_j, define a probability function $\mu_{\text{prop},k}(\cdot, P_j) : Outc \to 2^{Outc}$ as follows,

$$\mu_{\text{prop},k}(P_i, P_j) = \frac{CM_{\text{prop},k}[P_i, P_j]}{\sum_{l=1}^{|2^{\mathcal{P}}|} CM_{\text{prop},k}[P_l, P_j]}, \quad \forall P_i \in 2^{\mathcal{P}}, \tag{4}$$

where $CM_{\text{prop},k}[P_i, P_j]$ is the element of the confusion matrix $CM_{\text{prop},k}$ with predicted class label P_i and true class label P_j. That is, for every confusion matrix $CM_{\text{prop},k}$ where $k \in \{1, \ldots, k_{\text{max}}\}$, we define a total of $2^{|\mathcal{P}|}$ different probability functions, one for each possible true environmentP_j. Thus, the probability function $\mu_{\text{prop},k}$ that characterizes the probability of detecting an environment satisfying propositions P_i, given that the true environment at z_k satisfies propositions P_j. This helps to formally define the state transition probability of the overall system as follows.

Definition 3 (Transition probability function for proposition-labeled confusion matrices)**.** Let x_e be the true environment state corresponding to propositions P_j evaluating to true, and let $s_{a,1}, s_{a,2} \in S$ be states of the car. Let $O(s_1, s_2)$ denote the set of all predictions of the environment that prompt the system to transition from $s_1 = (s_{a,1}, x_e)$ to $s_2 = (s_{a,2}, x_e)$. At state s_1, let z_k be the distance interval of objects in the environment causing the agent to transition from $s_{a,1}$ to $s_{a,2}$. The corresponding confusion matrix is $CM_{\text{prop},k}$. Then, the transition probability from state s_1 to s_2 is defined as follows,

$$Pr(s_1, s_2) := \sum_{P_i \in O(s_1, s_2)} \mu_{\text{prop},k}(P_i, P_j). \tag{5}$$

For simplicity, we assume that objects at a specific distance interval influence the agent to transition from $s_{a,1}$ to $s_{a,2}$. However, Definition 3 can be extended to cases in which objects at multiple distances can influence transitions.

B. Class-labeled, distance-parametrized Confusion Matrix

This performance metric builds on the class-labeled confusion matrix defined in Definition 2. As denoted previously, let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be the set of different classes of objects in dataset $\mathcal{D}$. For every object in $\mathcal{D}_k$, the predicted class of the object will be one of the class labels $c_1, \ldots, c_n$. For each distance interval z_k, we define the class-labeled confusion matrix as $CM_{\text{class},k} := CM(\mathcal{C}, \mathcal{E}_k, \mathcal{D}_k)$. Algorithm 2 shows the construction of the class-labeled, distance-parametrized confusion matrix. Therefore, the outcomes of the object detection algorithm will be defined by the set $Outc = \{c_1, \ldots, c_n\}^m$, where m is the total number of objects in the true environment in the distance interval z_k. The tuple $(Outc, 2^{Outc})$ forms a σ-algebra for defining a probability function over the class-labeled confusion matrix $CM_{\text{class},k}$. Similar to the definition of a probability function, for every class label c_j, the probability function $\mu_{\text{class},k}(\cdot, c_j) : Outc \to [0,1]$ is defined as follows,

$$\mu_{\text{class},k}(c_i, c_j) := \frac{CM_{\text{class},k}[c_i, c_j]}{\sum_{l=1}^{n} CM_{\text{class},k}[c_l, c_j]}. \tag{6}$$

Algorithm 2 Class-labeled Confusion Matrix

procedure `ClassCM`(Dataset $\mathcal{D} = \{(f_i, b_i, d_i, x_i)\}_{i=1}^{N}$, Classes $\mathcal{C}$, Distance Parameters $\{D_k\}_{k=0}^{k_{\text{max}}}$)
  From $\{D_k\}_{k=0}^{k_{\text{max}}}$, define distance intervals $\{z_k\}_{k=1}^{k_{\text{max}}}$
  Run object detection algorithm to get predictions $\mathcal{E}$,
  Initialize $\mathcal{D}_1, \ldots, \mathcal{D}_{k_{\text{max}}}$ as empty sets
  Initialize $\mathcal{E}_1, \ldots, \mathcal{E}_{k_{\text{max}}}$ as empty sets
  for $(f_i, b_i, d_i, x_i) \in \mathcal{D}$ **do**
    if $d_i \in z_k$ **then**
      $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{(f_i, b_i, d_i, x_i)\}$
      $\mathcal{E}_k \leftarrow \mathcal{E}_k \cup \{(b_i, \tilde{x}_i)\}$
  for $k \in \{0, \ldots, k_{\text{max}}\}$ **do**
    Denote $CM_{\text{class}}(\mathcal{C}, \mathcal{E}_k, \mathcal{D}_k)$ as $CM_{\text{class},k}$
    $CM_{\text{class},k} \leftarrow$ zero matrix
    for $f_i \in \{f_1, \ldots, f_m\}$ **do** ▷ Loop over images
      for object in $\mathcal{D}_k$ **do**
        $c_i \leftarrow$ Predicted class label of object
        $c_j \leftarrow$ True class label of object in $\mathcal{E}_k$
        $CM_{\text{class},k}(c_i, c_j) \leftarrow CM_{\text{class},k}(c_i, c_j) + 1$
  $CM_{\text{class}}(\mathcal{C}, \mathcal{E}, \mathcal{D}) = \{CM_{\text{class}}(\mathcal{C}, \mathcal{E}_k, \mathcal{D}_k)\}_{k=0}^{k_{\text{max}}}$
  return $CM_{\text{class}}(\mathcal{C}, \mathcal{E}, \mathcal{D})$

Definition 4 (Transition probability function for class-labeled confusion matrix)**.** Let the true environment be represented as a tuple x_e corresponding to class labels in the region z_k (class labels can be repeated in a tuple x_e when multiple objects of the same class are in region z_k). Let $s_{a,1}, s_{a,2} \in S$ be states of the car, and let $O(s_1, s_2)$ denote the set of all predictions of the environment that prompt the system to transition from $s_1 = (s_{a,1}, x_e)$ to $s_2 = (s_{a,2}, x_e)$. Likewise, the tuple y_e represents the object detection model's predictions of the environment. Then, the transition probability function from state s_1 to s_2 is defined as follows,

$$Pr(s_1, s_2) := \sum_{y_e \in O(s_1, s_2)} \prod_{i=1}^{|y_e|} \mu_{\text{class},k}(y_e(i), x_e(i)). \tag{7}$$

For both transition probability functions (7) and (5), we can check (by construction) that $\forall s_1 \in S, \sum_{s_2} Pr(s_1, s_2) = 1$. In the running example, if the crosswalk were to have another pedestrian and a non-pedestrian obstacle, then the

probability of detecting each object is considered independently of the others. This results in the product of probabilities $\mu_{\text{class},k}(\cdot, x_e(i))$ in equation (7).

C. Markov Chain Construction [20]

For each confusion matrix, we can synthesize a corresponding Markov chain of the system state evolution as per Algorithm 3. As a result of prior work shown in [20], using off-the-shelf probabilistic model checkers such as Storm [39], we can compute the probability that the trace of a system satisfies its requirement, $\mathbb{P}(s_0 \models \varphi)$, by evaluating the probability of satisfaction of the requirement φ on the Markov chain. Let $O(x_e)$ be the set of all possible predictions of true environment state x_e by the the object detection model. The system controller $K : S \times O(x_e) \to S$ accepts as inputs the current state of the agent and the environment, $s_0 \in S$, and the environment state predictions $y_e \in O(x_e)$ from object detection. Based on the predictions, it actuates the agent resulting in the end state $s_f \in S$. At each time step, the agent makes a new observation of the environment (y_e) and chooses a control action corresponding to y_e.

Remark 4. Markov chain construction aids in evaluating the overall system. In future work, we plan to address the issue of tracking, in which perception errors are tracked over multiple temporal frames.

Algorithm 3 Markov Chain Construction

procedure `Markov Chain`$(S, K, O(x_e), CM, x_e)$
  $Pr(s, s') = 0, \forall s, s' \in S$
  for $s_o \in S$ **do**
    $\iota_{init}(s_0) = 1$
    for $y_e \in O(x_e)$ **do**
      $s_f \leftarrow K(s_o, y_e)$ ▷ Controller
      Identify z_k according to Definitions 3, 4
      $\mu_{\text{prop},k}, \mu_{\text{class},k} \leftarrow$ Equations (4), (6).
      if proposition-labeled **then**
        $P_j \leftarrow$ Propositions for true x_e
        $P_i \leftarrow$ Propositions for predicted y_e
        $p \leftarrow \mu_{\text{prop},k}(P_i, P_j)$
      if class-labeled **then**
        $p \leftarrow \prod_{i=1}^{|y_e|} \mu_{\text{class},k}(y_e(i), x_e(i))$
      $Pr(s_o, s_f) \leftarrow Pr(s_o, s_f) + p$
  return $\mathcal{M} = (S, Pr, \iota_{init}, AP, L)$

Proposition 1. Suppose we are given: i) φ as a temporal logic formula over system states S, ii) true state of the environment x_e, iii) agent initial state $s_{a,0}$, and iv) a Markov chain $\mathcal{M}$ constructed via Algorithm 3 for the distance parameterized confusion matrices constructed from Algorithm 1 or Algorithm 2, then $\mathbb{P}(s_0 \models \varphi)$ is equivalent to computing $\mathbb{P}_{\mathcal{M}}(s_0 \models \varphi)$, where $s_0 = (s_{a,0}, x_e)$.

V. Simulation Results

We present the distance-parametrized, proposition-labeled and class-labeled confusion matrices for a pre-trained YoLov3 model [22] evaluated on the first 85 scenes of the NuScenes dataset [23]. For this YoLov3 model and the controller described in the running example, we compute the probability of satisfaction that the car will meet its safety requirements given in equations (1)-(3). The YoLov3 [22] model was trained on the MS COCO dataset [40]. Note that we chose to evaluate a model on an evaluation set different from the source of the training set. Each scene is 20 seconds long, with 3D object annotations made at 2 Hz for 23 different classes. All objects with nuScenes annotation "human" are clustered under the class *ped*, and all objects annotated as "vehicle", "static obstacle", and moving obstacle are annotated as *obs*. We use all 40 frames from the `CAM-FRONT` sensor in each scene to form our dataset $\mathcal{D}$. The LiDAR sweeps accompanying each scene provides distances of annotated objects from the ego vehicle. We also project the annotated 3D bounding boxes from nuScenes to 2D to match the predicted detections from the YoLov3 model. These evaluations are used to construct the (distance-parametrized) class-labeled and proposition-labeled confusion matrices from Algorithms 1 and 2 with 10m distance intervals with parameters $D_0 = 0$ and $D_{k_{max}} = 100$m. The class-labeled confusion matrix for objects less than 10 m from the autonomous vehicle is given in Table I and the proposition-labeled confusion matrix for objects less

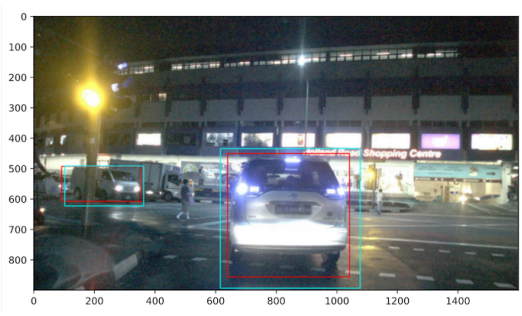


Fig. 3: YoLo-v3 detecting the car and truck as objects of class *obs*, but not detecting the pedestrians. The red boxes denote the YoLo predictions and the light blue boxes denote the corresponding ground truth annotations for correctly detected objects.

Pred \ True	*ped*	*obs*	*emp*
ped	31	0	0
obs	0	165	0
emp	121	665	2722

TABLE I: Class-labeled Confusion matrix for distance $d \leq 10.0$

Pred \ True	*ped*	*obs*	*ped, obs*	*emp*
ped	22	0	5	0
obs	0	158	4	0
ped, obs	0	0	0	0
emp	59	310	11	2722

TABLE II: Proposition-labeled Confusion matrix for distance $d \leq 10.0$. As expected, the sum total of samples for each label are no larger than for the corresponding label in the class-labeled confusion matrix.

than 10m from the autonomous vehicle is given in Table II. For brevity, the full distance-parametrized and proposition-labeled confusion matrices are at this GitHub repository[1]

For each confusion matrix and for each initialization of the car-pedestrian example, a Markov chain model of the overall system is constructed according to Algorithm 3, and the satisfaction probabilities for specifications (1)-(3) are computed using Storm [39]. As illustrated in Figure 4, the satisfaction probabilities of safety requirements are relatively low at around 20% for a maximum speed of 1. This is for several reasons. First, we evaluated a model trained on one modality (2D object detection with monocular vision); typically the best models are multi-modal and use data from several different sensors. Secondly, we do not consider tracking in our evaluation. Finally, we used a pre-trained model that was not trained on the NuScenes dataset in part because the objective of this work is to illustrate evaluation metrics and not necessarily to pick the best detection models. As a next step, we would like to conduct this analysis on object detection models trained on multi-modal data from LIDAR and multiple cameras, including baseline models from the nuScenes [23] detection challenge. The choice of a stronger object detection model would better highlight the strength of our evaluation framework, as illustrated in Figure 5. For each true positive rate for the pedestrian class, 50 random instances of the 3×3 class-labeled confusion matrix were generated. Even though the class-labeled confusion matrix is the most conservative, we observe that system-level satisfaction probability is close to 1 when the true positive rate is high ($> 99\%$).

Despite the low probabilities, we observe insightful differences in the satisfaction probabilities resulting from the choice of confusion matrix. Between canonical class based confusion matrix and its distance-parametrized counterpart (see Figures 4a and 4b), we see a two-fold increase in satisfaction probability $\mathbb{P}_{\mathcal{M}}(s_0 \models \varphi)$ for low speeds. Similar trends hold for for proposition-labeled confusion matrix and its distance parameterized variants as seen in Figures 4c and 4d. Across all confusion matrices, satisfaction probability decreases with speed, corresponding to not being able to recover from misdetections at higher speed, which is due to our choice of controller. Lastly, the proposition-labeled confusion matrix results in higher satisfaction probabilities than its class-labeled counterpart.

VI. Conclusion and Future Work

We introduced two evaluation metrics, a *proposition-labeled* confusion matrix and a *class-labeled distance parameterized* confusion matrix, for object detection tasks. Parameterizing the confusion matrix with distance accounts for differences in detection performance in the temporal logic analysis with the planning module. We empirically observed that the proposition-labeled confusion matrix resulted in less conservative satisfaction probabilities than the canonical class-labeled confusion matrix. The distance-parametrized metric further reduced conservativeness by accounting for variations in detection performance based on distance.

We envision several exciting directions for future work. First, to augment our framework to handle dynamic environments, we consider building on the work in [27], which studies quantitative analysis of systems that operate in partially known dynamic environments. It assumes that the environment model belongs to a set $\mathbf{M}^{env}$ of Markov chains. The system does not know the true model of the environment, and instead maintains a belief, which is defined as a probability distribution over all possible environment models in $\mathbf{M}^{env}$. We will extend our work to derive the belief update function based on the perception performance. Second, we plan on hardware demonstrations to validate the results of the quantitative analysis presented here. Finally, we would like to handle tracking to further reduce the conservativeness in our analysis.

[1] https://github.com/abadithela/Dist-ConfusionMtrx

References

[1] O. Koyejo, N. Natarajan, P. Ravikumar, and I. S. Dhillon, "Consistent multilabel classification." in *NeurIPS*, vol. 29, 2015, pp. 3321–3329.

[2] X. Wang, R. Li, B. Yan, and O. Koyejo, "Consistent classification with generalized metrics," *arXiv preprint arXiv:1908.09057*, 2019.

[3] B. Yan, S. Koyejo, K. Zhong, and P. Ravikumar, "Binary classification with karmic, threshold-quasi-concave metrics," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5531–5540.

[4] H. Narasimhan, H. Ramaswamy, A. Saha, and S. Agarwal, "Consistent multiclass algorithms for complex performance measures," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2398–2407.

[5] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang Jr, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[6] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[7] M. Lahijanian, S. B. Andersson, and C. Belta, "A probabilistic approach for control of a stochastic system from LTL specifications," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 2236–2241.

[8] 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*, 2011, pp. 313–314.

[9] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

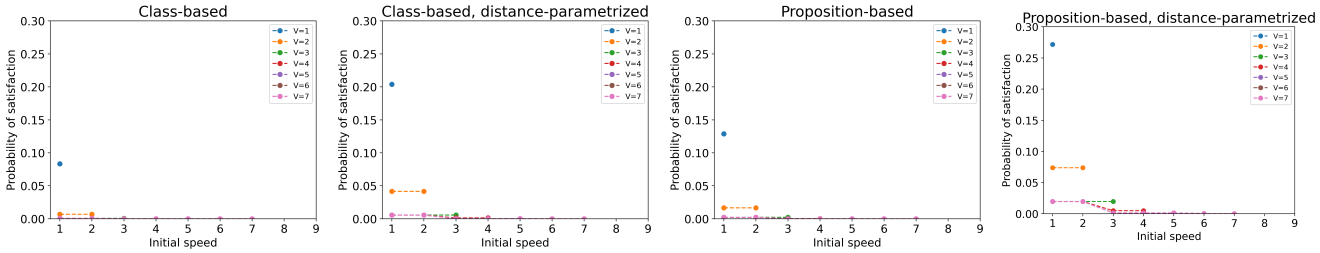
[10] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 81–87.

[11] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT press, 2008.

[12] J. Karlsson and J. Tumova, "Intention-aware motion planning with road rules," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 526–532.

[13] J. Karlsson, S. van Waveren, C. Pek, I. Torre, I. Leite, and J. Tumova, "Encoding human driving styles in motion planning for autonomous vehicles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1050–1056.

[14] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos, "Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic," in *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, 2019, pp. 1–11.



(a) (b) (c) (d)

Fig. 4: Satisfaction probabilities $\mathbb{P}_{\mathcal{M}}(s_0 \models \varphi)$ for the car pedestrian example with Markov chains $\mathcal{M}$ derived from the various confusion matrices. Each plot shows the satisfaction probability as a function of initial speed, with maximum speed as the legend.

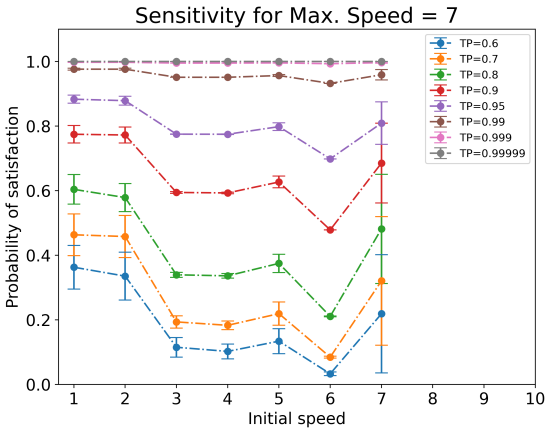


Fig. 5: Sensitivity analysis of the system-level satisfaction probability for varying true positive rates of detecting pedestrians. The errorbars report standard deviation on satisfaction probability for the randomly generated confusion matrices.

[15] B. Gassmann, F. Oboril, C. Buerkle, S. Liu, S. Yan, M. S. Elli, I. Alvarez, N. Aerrabotu, S. Jaber, P. van Beek *et al.*, "Towards standardization of AV safety: C++ library for responsibility sensitive safety," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2265–2271.

[16] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[17] T. Dreossi, S. Jha, and S. A. Seshia, "Semantic adversarial deep learning," in *International Conference on Computer Aided Verification*. Springer, 2018, pp. 3–26.

[18] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[19] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[20] A. Badithela, T. Wongpiromsarn, and R. M. Murray, "Leveraging classification metrics for quantitative system-level analysis with temporal logic specifications," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 564–571.

[21] C. S. Pasareanu, R. Mangal, D. Gopinath, S. G. Yaman, C. Imrie, R. Calinescu, and H. Yu, "Closed-loop analysis of vision-based autonomous systems: A case study," *arXiv preprint arXiv:2302.04634*, 2023.

[22] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[23] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 621–11 631.

[24] B. Johnson and H. Kress-Gazit, "Probabilistic analysis of correctness of high-level robot behavior with sensor error," *Robotics: Science and Systems VII*, p. 137, 2012.

[25] S. Topan, K. Leung, Y. Chen, P. Tupekar, E. Schmerling, J. Nilsson, M. Cox, and M. Pavone, "Interaction-dynamics-aware perception zones for obstacle detection safety evaluation," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 1201–1210.

[26] P. Antonante, H. Nilsen, and L. Carlone, "Monitoring of perception systems: Deterministic, probabilistic, and learning-based fault detection and identification," *arXiv preprint arXiv:2205.10906*, 2022.

[27] T. Wongpiromsarn and E. Frazzoli, "Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012, pp. 7644–7651.

[28] S. Beland, I. Chang, A. Chen, M. Moser, J. Paunicka, D. Stuart, J. Vian, C. Westover, and H. Yu, "Towards assurance evaluation of autonomous systems," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–6.

[29] A. Dokhanchi, H. B. Amor, J. V. Deshmukh, and G. Fainekos, "Evaluating perception systems for autonomous vehicles using quality temporal logic," in *International Conference on Runtime Verification*. Springer, 2018, pp. 409–416.

[30] A. Balakrishnan, A. G. Puranic, X. Qin, A. Dokhanchi, J. V. Deshmukh, H. B. Amor, and G. Fainekos, "Specifying and evaluating quality metrics for vision-based perception systems," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1433–1438.

[31] H. Kress-Gazit, G. Fainekos, and G. Pappas, "Where's Waldo? Sensor-based temporal logic motion planning," in *Proc. of IEEE International Conference on Robotics and Automation*, 2007, pp. 3116–3121.

[32] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[33] D. Conner, H. Kress-Gazit, H. Choset, A. Rizzi, and G. Pappas, "Valet parking without a valet," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 572–577.

[34] A. Girard and G. J. Pappas, "Hierarchical control system design using approximate simulation," *Automatica*, vol. 45, no. 2, pp. 566–571, 2009.

[35] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, 2009.

[36] L. I. R. Castro, P. Chaudhari, J. Tumova, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 3217–3224.

[37] P. Tabuada and G. J. Pappas, "Linear time logic control of linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[38] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[39] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A Storm is coming: A modern probabilistic model checker," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 592–600.

[40] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.