Safety Assurance for Autonomous Systems with Multiple Sensor Modalities

Anand Balakrishnan*, Rohit Bernard*, Shreeram Narayanan*, Vidisha Kudalkar*, Yiqi Zhao*,
Parinitha Nagaraja†, Georgi Markov†, Christof Budnik†, Helmut Degen†,
Lars Lindemann* and Jyotirmoy V. Deshmukh*

***Abstract*—Humans and autonomous cyber-physical systems increasingly share physical space, for example, in industrial manufacturing, autonomous taxis, warehouses, and unmanned package delivery. This makes such autonomous CPS safety-critical because design errors can harm the people in their shared space. To enhance their own safe operation and the safety of humans around them, these CPSs typically use multiple sensor modalities to perceive the environment. Such sensor systems include RADAR, LIDAR, ultra-wideband, SONAR, odometry, GPS, and camera-based sensors to make estimations about their own state and observations of the environment. Traditionally, the observations made by different sensor streams are *fused* using probabilistic models such as Bayesian filters (e.g., Kalman filters). These filters make assumptions about the distribution of error between the observation and the ground truth for a given sensor and, using such assumptions, attempt to reconstruct a state estimate by computing some weighted combination of observations from multiple sensors (with possibly different error distributions). However, such assumptions can be challenging to model as environments become more complex.**

Furthermore, such algorithms typically do not account for sensor failures or shifts in the error distribution during deployment. This paper presents an algorithmic framework that defines a notion of *spatio-temporal consistency* across sensor streams. We eschew the idea of computing a fused state estimate and instead focus on producing a *consistent state estimate* if the multiple sensor observations are deemed consistent. If we detect an inconsistency in the state estimate, we propose a conservative over-approximation of the state estimate based on the last known consistent estimate. We demonstrate how such a framework can be deployed in an industrial manufacturing case study. We show that such a framework can provide probabilistic runtime assurance using conformal prediction techniques for statistical analyses.

I. Introduction

Many cyber-physical systems (CPSs) operate in shared spaces with humans, and the temporal behavior of the CPS or those of humans can evolve in a way that can cause physical harm to humans. For example, consider an industrial manufacturing robot; such robots can have high-momentum manipulator arms. An accidental collision with such a robotic arm can harm a factory floor worker navigating an area close to the robot. As another example, consider a warehouse where human operators must work alongside high momentum, autonomous mobile robots or forklifts. It is crucial that such robots also ensure the safety of their human co-workers.

In general, such multi-sensor systems have been designed with *sensor fusion* techniques to combine information from each individual sensor and reason about the environment on a joint representation derived them [1, 2, 3, 4]. Moreover, having an efficient and accurate sensor fusion pipeline is essential for any downstream decision-making to be viable, as erroneous data can lead to several potential failures, especially in safety-critical CPSs. Thus, any data fusion technique used for decision-making has to be robust erroneous sensor data, including dropped network packets, environmental noise, and adversarial environmental artifacts.

To this end, we propose the use of a *second-order* reasoning about the fused state estimate in a system: a notion of signal *consistency*. In this paper, we look at an industrial case study where we use an algorithmic framework where:

1) A monitoring algorithm that compares the information obtained from different sensor modalities and determines if the sensors collectively provide a *consistent estimate* of the environment state, or reports *inconsistency*.
2) The above allows us to inform a predictive monitoring algorithm that uses the sequence of determinations of consistency/inconsistency between the sensors to reason about the *reliability* of the various sensors and predict possible violations of the system's safety.

We then use conformal prediction [5, 6] as a statistical testing method for the correctness/conservativeness of our framework in detecting inconsistencies.

Related work: Prior work on defining a notion of conformance or "closeness" of trajectories in a system have relied on defining distance *metrics* between signals: a small such distance implies that the signals are spatially "close" together. This includes the Skorokhod distance [7, 8], the $(\tau-\epsilon)$-distance [9], dynamic time-warping distance [10], and other pseudo-distance measures [11].

Such distances work well for offline data, but are, in general, not feasible for fast online monitoring. In our work, we present a more general notion of "consistency" via a δ-spatial consistency metric, which takes distance between point-wise observations and performs a worst-case aggregate them over a monitoring window.

II. Preliminaries

Definition 1 (Dynamical System)**.** A dynamical system is defined as a tuple (S, A, F, π), where S is a compact set of states; A is a finite or infinite set of actions (that includes an empty action); and the transition dynamics F and the stochastic

2832-6520/24/$31.00 ©2024 IEEE
DOI 10.1109/MEMOCODE63347.2024.00018

control policy π are probability distributions. Let the state of the system at time t be denoted s_t and the action it takes at time t be sampled from π, denoted $a_t \sim \pi(a \mid s_t)$. Then, the transition dynamics is a conditional probability distribution over s_{t+1}, where $s_{t+1} \sim F(s' \mid s = s_t, a = a_t)$.

Let the set of non-negative reals, $\mathbb{R}^{\geq 0}$, denote the *time domain*. A *signal*, $\boldsymbol{z} : \mathcal{T} \to \mathcal{D}$, is a mapping from a finite subset of the time domain $\mathcal{T} \subset \mathbb{R}^{\geq 0}$ to some set $\mathcal{D}$. If $0 \notin \mathcal{T}$ for some $\boldsymbol{z}$, we say that $\boldsymbol{z}$ is a *partial signal or trajectory fragment*. For a given signal $\boldsymbol{z}$, we abuse notation to denote the time domain over which $\boldsymbol{z}$ is defined as $\mathcal{T}(\boldsymbol{z})$, and the *horizon* of a signal $\boldsymbol{z}$, $H(\boldsymbol{z}) = \max\{t \mid t \in \mathcal{T}(\boldsymbol{z})\}$, is the largest time point in the signal time domain. Given a time-domain $\mathcal{T}$, we use $C(\mathcal{T})$ to denote the convex closure of $\mathcal{T}$, i.e., if for every $0 \leq t_i, t_{i+1} \leq H(\mathcal{T})$, for all $\lambda \in [0, 1]$, $\lambda t_i + (1 - \lambda) t_{i+1} \in C(\mathcal{T})$.

Remark. Note that when we write $\boldsymbol{s}(t)$, for some $t \notin \mathcal{T}(\boldsymbol{s})$, we assume that the signal is interpolated (or extrapolated) using a mechanism such as constant or linear interpolation (or extrapolation), as appropriate for the state space S.

Definition 2 (State Trajectories as Signals)**.** Given an initial state $s_0 \in S$, the state trajectory is a signal $\boldsymbol{s} : \mathcal{T} \to S$ where the trajectory is sampled at discrete time points times $t_0, \ldots, t_T \in \mathcal{T}$ are mapped to $s_0, \ldots, s_T$, where $0 \leq t_i < t_{i+1} \leq t_T$, and $s_{i+1} \sim F(s' \mid s = s_i, a_t \sim \pi)$, for $i \in 0, \ldots, T-1$.

The (Markovian) controlled stochastic dynamical system often underlies real physical systems, e.g., an autonomous vehicle, robots navigating around humans, and humans in a shared space cohabited by manufacturing robots. However, a common feature of many real-world systems is that they lack full observability. As the state s_t of the system is not directly observable, a common design decision is to use a combination of sensors to estimate the actual state of the system.

Definition 3 (Observation space and Sensor signals)**.** An observation space for the dynamical system defined in Definition 1 is a tuple $(\mathcal{O}, Y)$, where $\mathcal{O}$ is a set of observations, and Y is a distribution on the space of observations conditioned on the current state. Given a state trajectory signal $\boldsymbol{s}$, the corresponding observation signal $\boldsymbol{o}$ is defined such that $\boldsymbol{o}(t) \sim Y(o \mid s = \boldsymbol{s}(t))$.

Essentially, an observation space gives us the space in which a sensor "observes" the underlying system state and the corresponding observation signal defines the time instants at which a sensor samples the system. Moreover, a given system can have multiple sensors observing it, and hence, we can have multiple observation spaces, each with its own sensor signals.

Signal Temporal Logic (STL) is a real-time logic, typically interpreted over signals that take values in a continuous metric space (such as $\mathbb{R}^m$) [12, 13]. We define the syntax for an STL formula φ through the following recursive grammar:

$$\begin{aligned}\varphi :=& \mu(\boldsymbol{z}(t)) > 0 \mid \neg\varphi \mid \varphi \wedge \varphi \\ & \mid \Box_I \varphi \mid \boxminus_I \varphi \mid \Diamond_I \varphi \mid \blacklozenge_I \varphi \\ & \mid \varphi \,\mathsf{U}_I\, \varphi \mid \varphi \,\mathsf{S}_I\, \varphi,\end{aligned}$$

where $\mu(\boldsymbol{z}(t))$ is a scalar function defined over the signal space, and I is an interval of the form $[a, b]$ for $a, b \in \mathbb{N}$. Here, the propositional logic operators for negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$) as defined as usual. Moreover, STL inherits the *temporal* operators from Linear Temporal Logic, whose Boolean satisfaction semantics can informally be defined as:

- $\Box_{[a,b]} \varphi$ says that φ must hold for all future samples in $[t + a, t + b]$, where t is the current time of evaluation, while $\boxminus_{[a,b]}$ is the past-time equivalent.
- $\Diamond_{[a,b]} \varphi$ says that φ must at least once in all future samples in $[t + a, t + b]$, and $\blacklozenge_{[a,b]}$ is the past-time equivalent.
- $\varphi_1 \,\mathsf{U}_{[a,b]}\, \varphi_2$ says that φ_1 must hold in $[t + a, t + b]$ until φ_2 holds, and $\varphi_1 \,\mathsf{S}_{[a,b]}\, \varphi_2$ is its past-time equivalent.

Conformal prediction was introduced in [5, 6] to obtain valid prediction regions for complex predictive models such as neural networks without making assumptions on the distribution of the underlying data. Let the *nonconformity scores* be a set of $k+1$ exchangeable real-valued randorm variables $R^{(0)}, \ldots, R^{(k)}$. In supervised learning, it is often defined as $R^{(i)} := \|Z^{(i)} - \mu(X^{(i)})\|$ where the predictor μ attempts to predict the output $Z^{(i)}$ based on the input $X^{(i)}$. Naturally, a large nonconformity score indicates a poor predictive model. The goal then is to obtain a prediction region for $R^{(0)}$ based on the calibration data $R^{(1)}, \ldots, R^{(k)}$, i.e., the random variable $R^{(0)}$ should be contained within the prediction region with high probability.

Formally, given a failure probability $\delta \in (0, 1)$, we want to construct a valid prediction region $C \in \mathbb{R}$ so that[1]

$$\text{Prob}(R^{(0)} \leq C) \geq 1 - \delta. \tag{1}$$

We pick $C := \text{Quantile}(\{R^{(1)}, \ldots, R^{(k)}, \infty\}, 1 - \delta)$ which is the $(1 - \delta)$th quantile of the empirical distribution of the values $R^{(1)}, \ldots, R^{(k)}$ and ∞. Equivalently, by assuming that $R^{(1)}, \ldots, R^{(k)}$ are sorted in non-decreasing order and by adding $R^{(k+1)} := \infty$, we can obtain $C := R^{(p)}$ where $p := \lceil (k + 1)(1 - \delta) \rceil$ with $\lceil \cdot \rceil$ being the ceiling function, i.e., C is the pth smallest nonconformity score. By a quantile argument, see [14, Lemma 1], one can prove that this choice of C satisfies (1). We remark that $k \geq \lceil (k + 1)(1 - \delta) \rceil$ is required to hold to obtain meaningful, i.e., bounded, prediction regions. It is known that the guarantees in (1) are marginal over the randomness in $R^{(0)}, R^{(1)}, \ldots, R^{(k)}$ as opposed to being conditional on $R^{(1)}, \ldots, R^{(k)}$.

Conformal prediction has been recently applied to Signal Temporal Logic runtime verification [15, 16], which assumes only partial realizations of the system and sensor trajectories and involves a time-series predictor. In our application, we

[1]More formally, we would have to write $C(R^{(1)}, \ldots, R^{(k)})$ as the prediction region C is a function of $R^{(1)}, \ldots, R^{(k)}$. For this reason, the probability measure P is defined over the product measure of $R^{(0)}, \ldots, R^{(k)}$.

consider the *offline* data $\boldsymbol{x} := (\boldsymbol{s}, \boldsymbol{o}) \in \mathcal{T} \to \mathcal{S} \times Y$, a concatenated signal of system evolution and sensor observations for the purpose of offline verification. Specifically, we consider the STL robust semantics $\rho(\phi, \boldsymbol{x}, t_0)$ [17, 18], which denote how robustly the trajectory $\boldsymbol{x}$ satisfies the STL specification ϕ with the start time t_0 (which is 0 in our application). Thus, we define the calibration nonconformity score $R^{(i)} := -\rho(\phi, \boldsymbol{x}^{(i)}, \tau_0)$, where $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k)}$ are the calibration trajectories, and the test nonconformity score $R^{(0)} := -\rho(\phi, \boldsymbol{x}, \tau_0)$, where we do not have access to the test trajectory $\boldsymbol{x}$ until test time. Following Eq. 1, we can see that $\text{Prob}(\rho(\phi, \boldsymbol{x}, \tau_0) \geq -C) \geq 1 - \delta$. Now, if $C < 0$, we conclude that $\text{Prob}((\boldsymbol{x}, \tau_0) \models \phi) \geq 1 - \delta$.

III. Case study: Industrial Safety

In this case study, we study a synthetic recreation of an industrial environment in a Unity-based simulator, where human operators/workers can move around the floor of a manufacturing factory where fixed manipulator robots are in operation. Here, an ultra-wideband radio-based localization system is used to position the human operators wearing unique tags on their person, along with a camera-based perception system to redundantly compute similar localization information.

Real-time Locating System (RTLS): An RTLS systems usually consist of multiple transponders (or tags) and multiple receivers (or anchors) in a space that are used to localize wireless tags in a space. Depending on the hardware, several algorithms can be used for localization, but these algorithms can result in localization errors in the presence of electromagnetic or reflective noise, or if a tag doesn't have direct line-of-sight with multiple anchors.

In our experiments, we simulate errors in the RTLS system by outputting a Gaussian noise perturbation of the true location of each object in the simulator. The parameters of the noise model are derived from the accuracy and precision specifications of RTLS systems currently used in similar manufacturing floors.

Perception-based Object Detection System: The perception system uses a single-stage monocular object detection model called SMOKE [19] to predict oriented 3D bounding boxes of objects in view of the camera, and is trained on a dataset from our simulator. The output of such models may contain localization and classification errors due to various factors, including occlusion of an object and insufficient training data.

A. Spatial Consistency Checking

In our system, each message from the RTLS and perception systems may consist of multiple detected objects. For each object with unique ID i, let $\boldsymbol{o}_{i,\text{rtls}}$ denote the location of i in RTLS observation signal with observation space $(\mathcal{T}_1, O_1, Y_1)$ for that object. Similarly, let $\boldsymbol{o}_{i,\text{vision}}$ denote the perception-based location in observation space $(\mathcal{T}_2, O_2, Y_2)$ for that object.

Now, we define aconsistency that is able to:

- operate on signals in observation spaces generated by significantly different sensor systems;
- reason about signals with different sampling frequencies; and
- output consistency information at each time point in the signal while being robust to outliers and missing data.

A *monitoring epoch* $T_\Delta \subset C(\mathcal{T}_1) \cap C(\mathcal{T}_2) \subset \mathbb{R}^{\geq 0}$ is a time interval over which we aggregate a *consistency metric* $\mathsf{CC}(\cdot, \cdot)$:

$$\mathsf{CC}(o_{i,\text{rtls}}, o_{i,\text{vision}}) = \|o_{i,\text{rtls}} - o_{i,\text{vision}}\|, \tag{2}$$

where $o_{i,\text{rtls}}, o_{i,\text{vision}}$ are samples from $\boldsymbol{o}_{i,\text{rtls}}$ and $\boldsymbol{o}_{i,\text{vision}}$. For a monitoring epoch T_Δ, a *consistency monitor* outputs for time t in the run of the system using Algorithm 1 such that

$$\mathsf{CC}_{T_\Delta}(\boldsymbol{z}_1, \boldsymbol{z}_2, t, \delta) = \forall t' \in [\tau, t]\,(\mathsf{CC}(z_1(t'), z_2(t')) \leq \delta)\,, \tag{3}$$

where $\tau = \max(0, t - T_\Delta)$. We choose the parameter $\delta := 0.05$, to be enforced that the signals for each subject in the scene are within 0.05m of each other; and choose $T_\Delta := 0.5$s with a period of 0.02s.

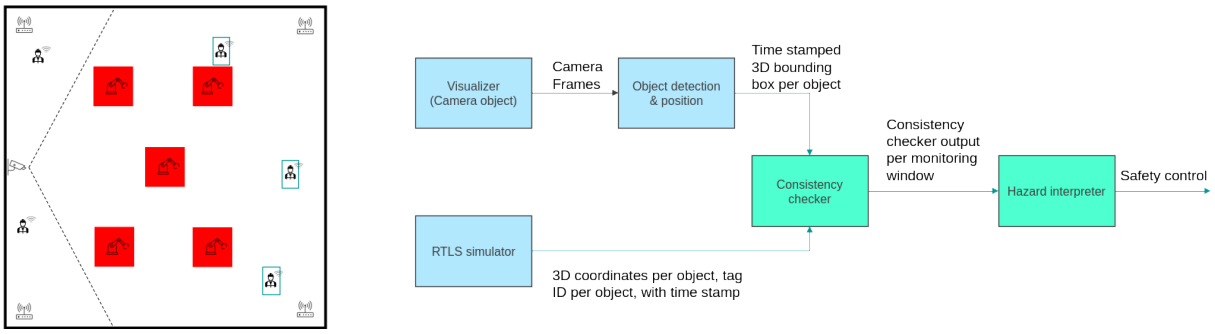
Algorithm 1 The procedure to compute the consistency output for two signals $\boldsymbol{z_1}, \boldsymbol{z_2}$ at time t for a monitoring epoch T_Δ.

procedure $\mathsf{CC}_{T_\Delta}(\boldsymbol{z_1}, \boldsymbol{z_2}, t, \delta)$
  $\tau = \max(0, t - T_\Delta)$
  $(i_0, \ldots, i_k) \leftarrow \mathcal{T}(\boldsymbol{z_1}) \cap [\tau, t]$
    ▷ i maintains the sample times for $\boldsymbol{z_1}$ in the epoch.
  $(j_0, \ldots, j_l) \leftarrow \mathcal{T}(\boldsymbol{z_2}) \cap [\tau, t]$
    ▷ j maintains the sample times for $\boldsymbol{z_2}$ in the epoch.
  $\boldsymbol{c}$ ▷ Signal maintaining consistency output for each time point.
  $t' = \max(i_0, j_0)$
  while $i < k$ and $j < l$ **do**
    $\boldsymbol{c}(t') \leftarrow \mathsf{CC}(\boldsymbol{z_1}(i), \boldsymbol{z_2}(j))$
    if $\min(k, i+1) \leq \min(l, j+1)$ **then**
      $i \leftarrow \min(k, i+1); t' \leftarrow i$
    else
      $j \leftarrow \min(k, j+1); t' \leftarrow j$
    end if
  end while
  if $\forall t' \in \mathcal{T}(\boldsymbol{c}), \boldsymbol{c}(t') \leq \delta$ **then**
    return $\mathsf{True}, \boldsymbol{c}$
  else
    return $\mathsf{False}, \boldsymbol{c}$
  end if
end procedure

Here, we see that by simply finding a 1-1 correspondence between objects in each stream that are within δ of each other, we naturally derive a consistency metric, which can be consumed by a downstream controller or "hazard interpreter".

B. Consistency-informed Safety Controller

In this study, we consider a conservative, fail-safe controller that prevents potential violations of the virtual fences around each robot (as depicted in Figure 1). The controller observes when a human operator violates (or is about to violate) the virtual fence of any particular robot, and stops the corresponding robot to prevent any potential harm to the person, using the following logic:



(a) Top-down Map (b) Pipeline

Figure 1. An industrial case study: A manufacturing factory floor where human operators share spaces with robots with virtual "fences" surround them. The objects in the scene are tracked via a camera-based object tracker, and the human operators wear tags that can be localized by a ultra-wideband localization system. The system consists of multiple sensor streams that are monitored by the consistency checker node.

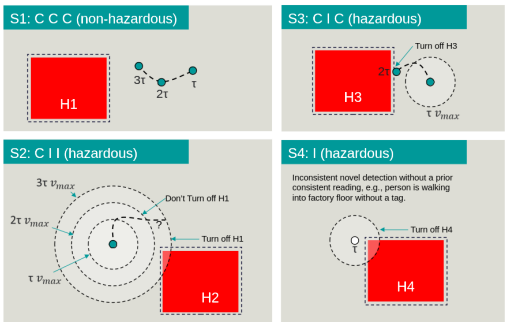


Figure 2. Conservative prediction of state from last known consistent location. In each scenario, the controller takes a short signal from the consistency checker (where C corresponds to a consistent reading, and I corresponds to inconsistent) and determines if it is hazardous or not.

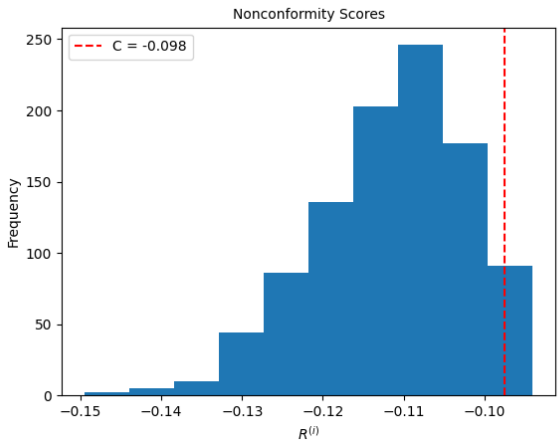


Figure 3. Nonconformity Scores $R^{(1)}, \ldots, R^{(K)}$. Here, the dotted line represents the $C = -0.098$ conformance, which allows us to conclude that the Prob$((x, \tau_0) \models \phi) \geq 1 - \delta$, for our particular choice of $\delta := 0.05$.

- **Consistent**: use the nominal policy π.
- **Inconsistent**: here, we use a worst-case estimate of how fast a person can move within a single controller period, $v_{\max}$. This allows us to extrapolate the worst-case position of any given *inconsistent* object from its last known consistent position, as seen in scenario 2 of Figure 2.

C. Verification

To verify the correctness of our choice in δ, we generate a representative dataset from our simulator with 5 human operators and 5 robots (as seen in Figure 1). Each trajectory in the dataset is generated by planning collision-free, random paths for each human operator across the factory floor, thereby covering various positional scenarios. Each trajectory is 60 seconds long, with the cameras sampled at 200 millisecond periods, and the RTLS data sampled at 20 millisecond periods.

We define the consistency specification for each object i in the system to be

$$\phi_i := \boldsymbol{c}_{T_\Delta, i}(t) \leq \delta \implies \boxminus_{[0,T_\Delta]}(\|\boldsymbol{s}(t) - \boldsymbol{o}_i(t)\| \leq 0.8), \quad (4)$$

Since we need the specification to apply for all objects in the scene, we define the global consistency specification to be $\min_i \rho(\phi_i, (\boldsymbol{s}, \boldsymbol{o}_o), 0)$.

We fix $\delta := 0.05$ and show the resulting nonconformity scores in Figure 3 with 1000 calibration samples. As we see, since $C < 0$, we can conclude that that Prob$((x, \tau_0) \models \phi) \geq 1 - \delta$.

IV. Conclusion

In this paper, we aim to tackle the problem of multi-sensor data fusion in the presence of unreliable or faulty sensor systems. Specifically, we present a notion of δ-spatial consistency, which allows us to extend point-wise distances between observations from multiple sensors to *spatiotemporal consistency*. We present an algorithm to determine consistency over monitoring windows given a sufficiently representative metric, along with a threshold below which a set of observations are considered "inconsistent". We show how to verify such a system, and demonstrate the framework in a real-world industrial use case, where consistency information is used to ensure safety in a mixed-autonomous environment.

References

[1] J. Hackett and M. Shah, "Multi-Sensor Fusion: A Perspective," in *, IEEE International Conference on Robotics and Automation Proceedings*, May 1990, pp. 1324–1330 vol.2.

[2] R. Gravina, P. Alinia, H. Ghasemzadeh, and G. Fortino, "Multi-Sensor Fusion in Body Sensor Networks: State-of-the-art and Research Challenges," *Information Fusion*, vol. 35, pp. 68–80, May 2017.

[3] L. A. Klein, *Sensor and Data Fusion: A Tool for Information Assessment and Decision Making*, 2nd ed., ser. PM. Bellingham, Wash: SPIE Press, 2007, no. 138.

[4] M. E. Liggins, D. L. Hall, and J. Llinas, Eds., *Handbook of Multisensor Data Fusion: Theory and Practice*, 2nd ed. Boca Raton, FL: CRC Press, 2017.

[5] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

[6] G. Shafer and V. Vovk, "A Tutorial on Conformal Prediction." *Journal of Machine Learning Research*, vol. 9, no. 3, 2008.

[7] R. Majumdar and V. S. Prabhu, "Computing the Skorokhod Distance between Polygonal Traces," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '15. New York, NY, USA: Association for Computing Machinery, Apr. 2015, pp. 199–208.

[8] J. V. Deshmukh, R. Majumdar, and V. S. Prabhu, "Quantifying Conformance Using the Skorokhod Metric," *Formal Methods in System Design*, vol. 50, no. 2-3, pp. 168–206, Jun. 2017.

[9] H. Abbas, H. Mittelmann, and G. Fainekos, "Formal Property Verification in a Conformance Testing Framework," in *2014 Twelfth ACM/IEEE Conference on Formal Methods and Models for Codesign (MEMOCODE)*. Lausanne, Switzerland: IEEE, Oct. 2014, pp. 155–164.

[10] M. Müller, "Dynamic Time Warping," in *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 69–84.

[11] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental Comparison of Representation Methods and Distance Measures for Time Series Data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, Mar. 2013.

[12] O. Maler and D. Nickovic, "Monitoring Temporal Properties of Continuous Signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, ser. Lecture Notes in Computer Science, Y. Lakhnech and S. Yovine, Eds. Berlin, Heidelberg: Springer, 2004, pp. 152–166.

[13] O. Maler, D. Nickovic, and A. Pnueli, "Real Time Temporal Logic: Past, Present, Future," in *Formal Modeling and Analysis of Timed Systems*, ser. Lecture Notes in Computer Science, P. Pettersson and W. Yi, Eds. Berlin, Heidelberg: Springer, 2005, pp. 2–16.

[14] R. J. Tibshirani, R. Foygel Barber, E. Candes, and A. Ramdas, "Conformal prediction under covariate shift," *Advances in neural information processing systems*, vol. 32, 2019.

[15] L. Lindemann, X. Qin, J. V. Deshmukh, and G. J. Pappas, "Conformal prediction for STL runtime verification," in *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*, 2023, pp. 142–153.

[16] Y. Zhao, B. Hoxha, G. Fainekos, J. V. Deshmukh, and L. Lindemann, "Robust conformal prediction for stl runtime verification under distribution shift," *arXiv preprint arXiv:2311.09482*, 2023.

[17] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[18] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[19] Z. Liu, Z. Wu, and R. Tóth, "SMOKE: Single-Stage Monocular 3D Object Detection via Keypoint Estimation," Feb. 2020.

[20] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," *arXiv preprint arXiv:2107.07511*, 2021.

[21] P. Antonante, D. I. Spivak, and L. Carlone, "Monitoring and Diagnosability of Perception Systems," *arXiv:2005.11816 [cs]*, May 2020.

[22] A. J. Barreto-Cubero, A. Gómez-Espinosa, J. A. Escobedo Cabello, E. Cuan-Urquizo, and S. R. Cruz-Ramírez, "Sensor Data Fusion for a Mobile Robot Using Neural Networks," *Sensors*, vol. 22, no. 1, p. 305, Dec. 2021.

[23] P. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[24] Y. Chen and G. Medioni, "Object Modelling by Registration of Multiple Range Images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, Apr. 1992.

[25] Z. Chen, "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond," *Statistics*, vol. 182, Jan. 2003.

[26] D. A. Dornfeld and M. DeVries, "Neural Network Sensor Fusion for Tool Condition Monitoring," *CIRP Annals*, vol. 39, no. 1, pp. 101–105, 1990.

[27] V. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian Filtering for Location Estimation," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24–33, Jul. 2003.

[28] L. H. Gilpin, C. Zaman, D. Olson, and B. Z. Yuan, "Reasonable Perception: Connecting Vision and Language Systems for Validating Scene Descriptions," in *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '18. New York, NY,

USA: Association for Computing Machinery, Mar. 2018, pp. 115–116.

[29] L. Gilpin, "Reasonableness Monitors," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[30] K. Kalpakis, D. Gada, and V. Puttagunta, "Distance Measures for Effective Clustering of ARIMA Time-Series," in *Proceedings 2001 IEEE International Conference on Data Mining*, Nov. 2001, pp. 273–280.

[31] L. I. Perlovsky and M. M. McManus, "Maximum Likelihood Neural Networks for Sensor Fusion and Adaptive Classification," *Neural Networks*, vol. 4, no. 1, pp. 89–102, Jan. 1991.

[32] J. Z. Sasiadek, "Sensor Fusion," *Annual Reviews in Control*, vol. 26, no. 2, pp. 203–228, Jan. 2002.

[33] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2043–2050.

V. Appendix

A. Semantics for STL

Formally, we define the Boolean satisfaction value of an STL formula, φ, for finite-length, discrete-time trace $\boldsymbol{z}$ at a given time instance t using the characteristic function $\beta(\varphi, \boldsymbol{z}, t)$, which is recursively defined as:

$$
\begin{aligned}
&\beta(\mu(\boldsymbol{z}(t)) > 0, \boldsymbol{z}, t) &&\iff \mu(\boldsymbol{z}(t)) > 0\\
&\beta(\neg\varphi, \boldsymbol{z}, t) &&\iff \neg\beta(\varphi, \boldsymbol{z}, t)\\
&\beta(\varphi_1 \wedge \varphi_2, \boldsymbol{z}, t) &&\iff \beta(\varphi_1, \boldsymbol{z}, t) \wedge \beta(\varphi_2, \boldsymbol{z}, t)\\
&\beta\big(\Box_{[a,b]}\,\varphi, \boldsymbol{z}, t\big) &&\iff \forall t' \in [t+a, t+b], \beta(\varphi, \boldsymbol{z}, t')\\
&\beta\big(\boxminus_{[a,b]}\,\varphi, \boldsymbol{z}, t\big) &&\iff \forall t' \in [t-b, t-a], \beta(\varphi, \boldsymbol{z}, t')\\
&\beta\big(\varphi_1\ \mathsf{U}_{[a,b]}\ \varphi_2, \boldsymbol{z}, t\big) &&\iff \exists t' \in [t+a, t+b], \beta(\varphi_2, \boldsymbol{z}, t')\\
&&&\qquad \wedge(\forall t'' \in [t', t), \beta(\varphi_1, \boldsymbol{z}, t''))\\
&\beta\big(\varphi_1\ \mathsf{S}_{[a,b]}\ \varphi_2, \boldsymbol{z}, t\big) &&\iff \exists t' \in [t-b, t-a], \beta(\varphi_2, \boldsymbol{z}, t')\\
&&&\qquad \wedge(\forall t'' \in [t', t), \beta(\varphi_1, \boldsymbol{z}, t''))
\end{aligned}
$$

We say that a signal $\boldsymbol{z}$ satisfies a formula φ at time t if $\beta(\varphi, \boldsymbol{z}, t)$ is true, denoted $(\boldsymbol{z}, t) \models \varphi$.

In addition to the Boolean satisfaction semantics, STL has quantitative semantics associated with it [18, 17]. The quantitative semantics define a *robustness* measure that computes the degree of satisfaction of an STL formula by a real-valued signal trace. Similar to the above Boolean satisfaction value definition, the robustness, $\rho(\varphi, \boldsymbol{z}, t)$, of the formula φ for the given signal $\boldsymbol{z}$ at time t is defined recursively as:

$$
\begin{aligned}
&\rho(\mu(\boldsymbol{z}(t) > 0, \boldsymbol{z}, t) &&= \mu(\boldsymbol{z}(t))\\
&\rho(\neg\varphi, \boldsymbol{z}, t) &&= -\rho(\varphi, \boldsymbol{z}, t)\\
&\rho(\varphi_1 \wedge \varphi_2, \boldsymbol{z}, t) &&= \min(\rho(\varphi_1, \boldsymbol{z}, t), \rho(\varphi_2, \boldsymbol{z}, t))\\
&\rho\big(\Box_{[a,b]}\,\varphi, \boldsymbol{z}, t\big) &&= \min_{t' \in [t+a, t+b]} \rho(\varphi, \boldsymbol{z}, t')\\
&\rho\big(\Diamond_{[a,b]}\,\varphi, \boldsymbol{z}, t\big) &&= \max_{t' \in [t+a, t+b]} \rho(\varphi, \boldsymbol{z}, t')\\
&\rho\big(\boxminus_{[a,b]}\,\varphi, \boldsymbol{z}, t\big) &&= \min_{t' \in [t-b, t-a]} \rho(\varphi, \boldsymbol{z}, t')\\
&\rho\Big(\diamondminus_{[a,b]}\,\varphi, \boldsymbol{z}, t\Big) &&= \max_{t' \in [t-b, t-a]} \rho(\varphi, \boldsymbol{z}, t')\\
&\rho\big(\varphi_1\ \mathsf{U}_{[a,b]}\ \varphi_2, \boldsymbol{z}, t\big) &&= \max_{\substack{t_1 \in\\ [t+a, t+b]}} \begin{Bmatrix} \rho(\varphi_2, \boldsymbol{z}, t_1),\\ \min_{t_2 \in [t, t_1)} \rho(\varphi_1, \boldsymbol{z}, t_2) \end{Bmatrix}\\
&\rho\big(\varphi_1\ \mathsf{S}_{[a,b]}\ \varphi_2, \boldsymbol{z}, t\big) &&= \max_{\substack{t_1 \in\\ [t-b, t-a]}} \begin{Bmatrix} \rho(\varphi_2, \boldsymbol{z}, t_1),\\ \min_{t_2 \in [t_1, t)} \rho(\varphi_1, \boldsymbol{z}, t_2) \end{Bmatrix}
\end{aligned}
$$