Model-Centered Assurance for Autonomous Systems

Susmit Jha, John Rushby(✉), and Natarajan Shankar

Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA
Susmit.Jha@sri.com, {Rushby,Shankar}@csl.sri.com

Abstract. The functions of an autonomous system can generally be partitioned into those concerned with perception and those concerned with action. Perception builds and maintains an internal model of the world (i.e., the system's environment) that is used to plan and execute actions to accomplish a goal established by human supervisors.

Accordingly, assurance decomposes into two parts: a) ensuring that the model is an accurate representation of the world as it changes through time and b) ensuring that the actions are safe (and effective), given the model. Both perception and action may employ AI, including machine learning (ML), and these present challenges to assurance. However, it is usually feasible to guard the actions with traditionally engineered and assured monitors, and thereby ensure safety, given the model. Thus, the model becomes the central focus for assurance.

We propose an architecture and methods to ensure the accuracy of models derived from sensors whose interpretation uses AI and ML. Rather than derive the model from sensors bottom-up, we reverse the process and use the model to predict sensor interpretation. Small prediction errors indicate the world is evolving as expected and the model is updated accordingly. Large prediction errors indicate surprise, which may be due to errors in sensing or interpretation, or unexpected changes in the world (e.g., a pedestrian steps into the road). The former initiate error masking or recovery, while the latter requires revision to the model. Higher-level AI functions assist in diagnosis and execution of these tasks.

Although this two-level architecture where the lower level does "predictive processing" and the upper performs more reflective tasks, both focused on maintenance of a world model, is derived by engineering considerations, it also matches a widely accepted theory of human cognition.

1 Introduction

Autonomous systems—those, such as self-driving cars, that are capable of independent decision making—typically use software that employs methods of Artificial Intelligence (AI) such as Machine Learning (ML) and time-constrained best-efforts deduction. It is generally impossible to predict the behavior of such software in all cases, as it can differ from case to case with little generalization. Consequently, it is difficult to provide assurance to justify confidence that an autonomous system is fit for use in critical contexts.

A. Casimiro et al. (Eds.): SAFECOMP 2020, LNCS 12234, pp. 228–243, 2020.
https://doi.org/10.1007/978-3-030-54549-9_15

One primitive way to seek assurance is through a large number of realistic tests. For self-driving cars, this is known as "collecting miles" and involves real and simulated driving through many scenarios and over large distances [1]. However, studies including one by the RAND Corporation [2], show that the quantity of testing required to substantiate acceptable safety is infeasibly vast, ranging from hundreds of millions to tens of billions of test miles.

An alternative that is endorsed for unmanned aircraft [3], for example, is to use conventional software as a monitor[1] for autonomous functions. The monitor is assured by conventional methods and overrides the autonomous software when it detects a potential safety violation and substitutes safe alternative behavior. Monitors have their own difficulties, however, but to examine them we first need to review the architecture of autonomous systems.

The functions of an autonomous system generally can be partitioned into those concerned with perception and those concerned with action. The purpose of perception is to build a model of the system's "world" (i.e., its environment) that is used for planning and executing actions to accomplish a goal established by human supervisors (e.g., the destination of a self-driving car) while maintaining required invariants, such as safety. This is an instance of model-based control, where the novelty is that the model is built with the aid of perception functions that employ AI and ML.

The overall case for safety depends on both perception (how good is the model of the world?) and action (how safe are the selected actions, given the model of the world?). Although AI and ML may be needed to construct an action plan, conventional software can check its safety, given a model of the world. Because it operates in uncertain environments (e.g., a self-driving car does not know what other road users may do) the checking software needs to consider all possibilities. Mobileye promote a safety model of this kind: "Responsibility-Sensitive Safety" (RSS) requires that the action software, and hence its monitor, should not be the cause of an accident no matter what other vehicles do [4]. With these interpretations, the action functions of an autonomous system can be checked or monitored by conventional software that can be assured by conventional means. However, the monitors also need a model of the environment and this complicates the argument for their assurance.

One possibility is that a monitor builds its own model, which might be very crude. For example, the monitor for a self-driving car might comprise a set of functions similar to those of an "Advanced Driver Assistance System" (ADAS) for human drivers, including "Forward Collision Warning," "Automatic Emergency Braking," "Blind Spot Warning," and "Rear Cross Traffic Warning and Rear Automatic Emergency Braking." Here, the "models" are derived very directly from radar and other sensors. These "last second" protections work well for human drivers but might be less effective for the unpredictable failures of autonomous systems. Furthermore, because they are unconnected to the larger plan constructed by action software, these protections may sometimes prove counterproductive. For example, the primary system might choose to give a

[1] The terms "guard," or "shield," or "safety bag" are also used.

cyclist or pedestrian a wide berth, but a “lane keeping assist” ADAS-like function operating as a monitor could override and steer back toward the cyclist or walker.

We can distinguish “independent” vs. “integrated” monitors at this point. An independent monitor uses its own sensing and decides whether or not to intervene without much reference to the primary action system. In addition to “last second” protections, a second function of an independent monitor could be to check whether the primary system is operating within specified “Operational Design Domains” (ODDs) [5]; these are circumstances, such as “on a freeway, in daylight,” for which the system was designed and is considered safe. In both cases, since it is performing different functions than the primary, it is plausible that such a monitor could provide strong protection using only assured conventional software in its sensing and interpretation (i.e., model-building) functions. We endorse such monitors for “within ODD” and “last second” protection, but prefer integrated monitors for overall assurance.

An integrated monitor evaluates the plan produced by action software, looking several time units ahead. To do this, it needs a model of the world and this model needs to resemble that of the primary system more than do the models of independent monitors. If an integrated monitor builds its own model using assured conventional software, then it seems this is likely to generate false alarms as it will be less accurate than the ML-enabled model of the primary system, or else it invites the question why the primary could not also use conventional software. On the other hand, if the monitor itself uses AI and ML software then we face the original problem of how this can be assured. And if the monitor does use AI and ML software, then a case can be made that it should use some or all of the same sensors and software as the primary system, because a redundant system will be expensive yet surely the recipient of less development and validation resources than the primary system, and therefore less capable and less trustworthy.

In the following, we will assume an integrated monitor that either uses the same model as the primary system or uses much the same sensors and software to build an alternative “shadow” model optimized for its different function. We will refer to the model used by the monitor, whether it be the primary model or its own specialized one, as the *assured model* because it is the focus for safety and therefore the architecture of the system should be directed to ensuring its accuracy. We call this “model-centered assurance” and it is the approach that we advocate. If the assured and primary models are separate, they both will be constructed and maintained in the way we describe below, but the details will be adjusted for their different functions.

The next section develops the idea of model-centered assurance and its key components: predictive processing and the dual-process architecture. Section 4 outlines the case for assurance, and Sect. 5 provides brief conclusions. The paper provides neither mathematics nor a case study. We are grateful to the organizers and reviewers for allowing us to present a purely conceptual paper. The underlying mathematics is well known in other fields and we provide references later; we are working on a case study and hope to report soon.

2 Model-Centered Assurance

If a world model is to be the focus for assurance, we need to consider the properties needed for safety, and the hazards to these. The intuitive property required of the model is fidelity to the real world, at least insofar as required to plan safe actions. In particular, if M is a model of world W, then any actions planned and shown to be safe using M should be safe when executed in W. The inverse need not be true; that is, there can be safe actions in the real world that do not appear so in the model. Thus, the model does not need to be a strictly accurate representation of the world: it can be conservative or, as we say, *safely approximate*.

In the absence of monitors, the model could be expressed in the variables of some latent space discovered by ML. But requirements for assurance monitors will surely be expressed in naturalistic terms ("don't hit a pedestrian") and the model must therefore represent these fairly directly. Hence, in model-centered assurance, the model must be a *naturalistic* one. Beyond that, the content of the model and submodels constructed by perception functions will be determined by the system and its purpose and goals. For a self-driving car, the overall model will represent something like an annotated map of its surroundings: it will include the layout of the local roads and junctions, information about nearby vehicles and pedestrians, locations of fixed objects (e.g., mailboxes), road signs and their interpretation, traffic signals, and so on. It is a design choice whether there is one model or a coordinated collection of submodels. A typical submodel is a list of detected objects in the vicinity of the *ego* vehicle, giving the location, size, velocity, type, and inferred intent of each; another might be an occupancy grid (a discretized probabilistic 2D map, describing the occupancy of each "square"); and another might be a representation of traffic lanes. Inputs from many different sensors (e.g., GPS location and map data, odometry, lidar, radar, proximity sensors, cameras etc.) will be interpreted and fused to create the world model and its submodels. All of these data sources present challenges in interpretation but we will focus on image data from cameras since this is the most widely studied and discussed, and among the most challenging.

Lin and colleagues [6] outline a representative vision processing pipeline for a self-driving car: image data from the cameras are fed in parallel to an object detector and a localizer; the object detector discovers the objects of interest around the vehicle, such as other vehicles, pedestrians, and traffic signals; the detected objects are then passed to an object tracker that associates objects with their past movements and predicts the future trajectories of moving objects [7]; in parallel, the localizer determines the location of the *ego* vehicle. Subsequently, the object movement information from the object tracker and vehicle location information from the localizer are combined and projected onto the same 2 or 3 dimensional coordinate space by a sensor fusion engine to create the model used by the action functions.

The object detector is usually based on (semi) supervised learning using a convolutional neural network. This is a deep learning neural network architecture that typically comprises some layers of artificial neurons arranged to perform convolution, some to do pooling, and some fully connected. The number and arrangement of layers of each type are determined by experiment and constitute the developer's "secret sauce." During subsequent training, a vast number of images are fed to the network together with correct (human generated) data on the objects they contain. During the training process, the weights in the neural network are adjusted until it detects objects with high accuracy and reliability.

Now although deep neural networks such as those employed in object detectors are astonishingly effective at their tasks, we must heed Judea Pearl's observation that at bottom all they are doing is extremely sophisticated curve fitting [8]. We tune the network architecture and its weights so that training data is detected and classified with high accuracy and reliability (but without "overfitting") and we hope that live data is detected equally well. This hope has to rest on the assumption that real world data is smooth in a way that matches the implicit curve of the neural network. But the lesson from safety-critical systems is that their failures are often associated with unanticipated cases that are discontinuous from their neighbors.

This concern is exemplified by so-called "adversarial examples" [9]. These are typically minor modifications to an input image that are indiscernible to a human observer but cause an image classifier to change its output, often drastically and inappropriately. Masks have been developed that will cause misclassification when overlayed on any image input to a given classifier, and there are universal examples that will disrupt any classifier [10]. Furthermore, there are patterns that can be applied to real world artifacts (e.g., small images that can be stuck to traffic signs) that will cause them to be misread by an object classifier [11]. There is much work on detection and defense against such attacks but their effectiveness is limited: for example, Carlini and Wagner [12] demonstrate that ten different methods for detecting adversarial examples are easily bypassed.

In our opinion these concerns and defenses miss the true significance of adversarial examples. Their significance is not that a classifier can be mislead by carefully manipulated inputs, but that they demonstrate that classifications are fragile: that is, for every correctly classified image, there may be several incorrectly classified ones just a few pixel changes away. And why wouldn't this be, when all that the learning-enabled components are doing is curve fitting? Shamir and colleagues present an analysis that confirms and quantifies this interpretation [13], while Kilburtus and colleagues argue that this kind of "anti-causal" learning via curve-fitting has inherent challenges in achieving generalization [14].

The inevitable conclusion is that although many learning-enabled systems driven by neural networks exhibit amazing performance, it is reckless to assume this performance is robust. These systems will always exhibit more failures than can be tolerated in safety-critical systems and this cannot be fixed by adjustments to the interpretation function because it derives from the fact that the learning-enabled software is not interpreting the world in any meaningful sense, but merely applying statistical correlation to patterns of pixels. The question then is: can we locate these learning-enabled components in an architecture or context where their failures can be detected and masked or tolerated?

Some such methods attempt to assess "confidence" in the quality of the current learning-enabled judgment. One way to do this is to estimate whether the current input is similar to those encountered during training. Pimentel *et al* review methods for doing this [15]. A popular modern method uses a variational autoencoder [16] to learn the training data distribution and to compress it to a smaller "latent" space, from which a decoder can reconstruct the original input. In live operation, encoding and decoding the current input will usually result in an image close to the original if it is similar to the training data, and distant if not. Measures such as "Mahalanobis distance" can be used to assess the distance between an image and its reconstruction. Another approach attempts to locate the portions of the input that most directly contribute to the current judgment and checks that these are related to the feature concerned [17]. We consider it prudent to employ methods such as these but, as with defenses against adversarial examples, we note that they can often be defeated by changes to a few pixels and do not contribute greatly to assurance.

A different class of methods employs diversity and fault tolerance. All fault tolerance is based on redundancy, but it requires more than mere replication. For example, some of the best performing learning-enabled systems for image and speech recognition employ "ensembles" in which several different systems cooperate or compete to generate results [18]. However, while this approach can improve already impressive behavior, it does not deliver assurance—because we have no reason to suppose that failures of the component systems are independent, and no reason to trust any individual system.

However, although diverse interpretations of the same sensors may not fail independently, it is plausible that different sensors, particularly different kinds of sensors, may do so. Hence, sensor fusion over different kinds of sensor may be able to detect or mask some failures, although this may not be straightforward to achieve, as illustrated by the fatal accident between an Uber self-driving car and a pedestrian in Arizona on 18th March 2018 [19]. The Uber car used three sensor systems to build a simple object tracker model (recall Lin's vision processing pipeline [6]): cameras, radars, and lidar. In each of these sensor systems, its own object detector indicates the position of each detected object and attempts to classify it as, for example, a vehicle, a pedestrian, a bicycle, or other. The object tracker fuses these inputs using a "prioritization schema that promotes certain tracking methods over others, and is also dependent on the recency of the observation" [19, page 8]. In the case of the Arizona crash, this resulted in a "flickering" identification of the victim as the sensor systems' own classifiers changed their identifications, and as fusion performed by the object tracker preferred first one sensor system, then another, as listed below [19, Table 1].

- 5.6 s before impact, victim classified as *vehicle*, by radar
- 5.2 s before impact, victim classified as *other*, by lidar
- 4.2 s before impact, victim classified as *vehicle*, by lidar
- Between 3.8 and 2.7 s before impact,
  classification alternated between *vehicle* and *other*, by lidar
- 2.6 s before impact, victim classified as *bicycle*, by lidar

- 1.5 s before impact, victim classified as *unknown*, by lidar
- 1.2 s before impact, victim classified as *bicycle*, by lidar

The deeper harm of this "flickering" identification is that "if the perception system changes the classification of a detected object, the tracking history of that object is no longer considered when generating new trajectories" [19, page 8]. Consequently, the object tracker never established a trajectory for the victim and the vehicle collided with her even though she had been detected in some form or other for several seconds.

There are two related problems here: one is that the object tracker maintains a rather impoverished model of the world, the other is that its method of sensor fusion and model update pays no attention to the prior state of the model. The purpose of perception in autonomous systems is to build a model that accurately reflects the world; the model therefore encodes a lot more information than does an individual sensor sample and we should be cautious about making large updates to the model on the basis of such an individual sample.

The tenet of Model-Centered Assurance is that the model is the focus of attention, and the goal of perception is to ensure that it is an accurate (or "safely approximate") representation of the world. When a sensor samples the world, its interpretation software delivers some representation and we need to decide how to incorporate this into the world model. On the one hand, the information from the sensor is new but, on the other, its interpretation is possibly erroneous. Furthermore, the model is the repository of much accumulated information, so the method of model update must maintain its overall accuracy and coherence.

These considerations suggest a radical alternative to the traditional relationship between sensors and model: instead of operating "bottom up," we invert this to derive a framework for perception known as *predictive processing* (PP).

2.1 Predictive Processing

The world causes the sense impressions that our sensors report, but standard bottom-up methods of sensor interpretation attempt to reason backwards or "anti-causally" from sense impressions to their causes. This is inherently fraught with difficulty because many different configurations in the world can cause the same sense impressions. The idea of predictive processing is to reason causally: if we have an accurate model of the world, we can use it to predict the impressions that the world will cause our sensors to perceive. This is valuable for several reasons. The first is that it assists the lower levels of sensor interpretation (which may still operate anti-causally) by telling them what to look for.

Consider the component of a car's vision system concerned with detecting traffic lanes. This can be accomplished by a computer vision algorithm (e.g., based on the Hough Transform [20]) that looks for more-or-less straight lines painted on the road. A bottom-up approach will perform this algorithm afresh as each camera frame is processed. But this is inefficient—the traffic lanes in the current image frame are likely to be similar to those in the previous few frames and we should surely use this to seed the search—and it is fragile, as missing or scuffed lane markers might cause lanes to go undetected where they could be detected if we knew where to look, or could even be extrapolated from previous images. A better approach builds a model of the road and its traffic lanes and projects this forward in time to guide the algorithm in its search for lanes in the current image by predicting their location. The lanes found by the algorithm will generally be very close to those predicted by the model, so the information returned by the vision system to the system that maintains the model could be just the *difference* between them: that is, the *prediction error*.

In general, there will be uncertainty in the world model. This might be represented as a collection of candidate models: for example, one model might suppose that a detected object is a bicycle, another that it is a pedestrian, and we will favor the one that has the least prediction error (bicycles generally go in the same direction as the traffic, while pedestrians tend to go across it, so the two models will predict different locations for the object in the next frame).

This is an example of "analysis by synthesis," meaning that we formulate hypotheses (i.e., candidate world models) and favor those whose (synthesized) predictions match the sensed data. Enabling this approach is the second reason why PP is so valuable. In practical applications, we need to consider the level of the predictions concerned: do we use the world model to synthesize the raw data (e.g., pixels) that we predict the sensor will detect (which can be done using autoencoders), or do we target some higher level of its local interpretation (e.g., detected objects)? The best choice is a topic for experimental research, but we would expect a level of representation comparable to that produced by an object detector to be suitable.

Rather than (or in addition to) multiple candidate models, we could use probability distributions to represent uncertainty about model parameters. Implementations of this approach often employ Bayesian methods, although there are many variations [21]. The component that maintains the model will then send its predictions to the vision (or other sensor) component as a *prior* distribution and the vision system will return a *posterior* one. This kind of Bayesian inference typically generates intractable integrals, so implementations employ methods known as *Variational Bayes* that turn the problem into iterative optimization of the posterior models to minimize prediction error [16].

There is strong similarity between this approach and a Kalman Filter. As here, a Kalman filter maintains a model of the system state, which is the estimated value of each of its state variables together with their estimated accuracies, and predicts their values at the next timestep. Sensors sample these values, and the state variables are updated in a way that favors their predicted values when these are believed to have high accuracy, and the sensor values otherwise.

In a further, and perhaps surprising similarity, the top-down approach is widely believed to be the way perception works in human (and other) brains, as first proposed by Helmholtz in the 1860s [22]. *Predictive Processing* [23], also known as predictive *coding* [24] and predictive *error minimization* [25], posits that the brain builds models of its environment and uses these to predict its sensory input, and that much of its activity can be seen as (an approximation to) iterative Bayesian optimization of its models to minimize prediction error. PP has prior "predictions" flowing from models down to sense organs and Bayesian "corrections" flowing back up that cause the posterior models to track reality. ("Free Energy" [26] is a more all-encompassing theory that includes actions: the brain "predicts" that the hand, say, is in a certain place and to minimize prediction error the hand actually moves there.) This is consistent with the fact that the brain has more neural pathways going from upper to lower levels than vice versa: models and predictions flow down, and only corrections flow up.

As we noted, PP can be seen as extending the idea of Kalman filtering from classical representations of state (i.e., a set of continuous variables, as in control theory) to more complex models, where we also have representations of object "type" and "intent" and so on. Thus, fields as varied as neuroscience, control theory, signal processing, and sensor fusion all employ methods similar to PP, but under different names and derived by different histories.

The model-centered architecture is portrayed in Fig. 1 (ignore the dark pink box on the right labeled "Level 2 model refinement" for the time being). The black arrows from "Level 1 model construction" to sensors and their low-level

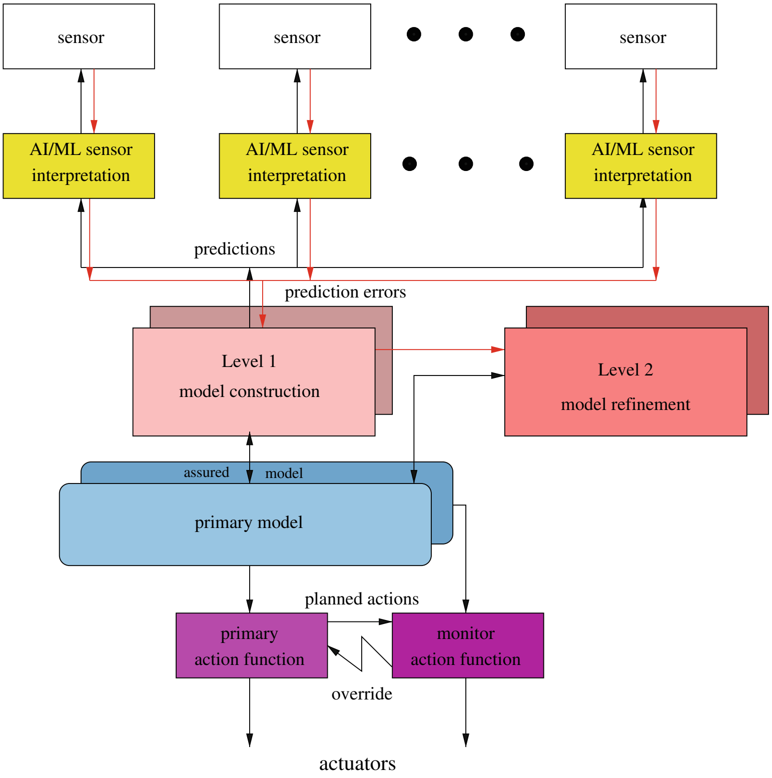

Fig. 1. Primary/monitor two-level architecture with predictive processing. (Color figure online)

interpretation indicate the top-down predictions of PP (inverted in the picture), while red arrows indicate prediction errors returned to model construction. If the assured model is separate from the primary model, then it and its mechanisms are indicated by the "shadows" behind the corresponding primaries.

The third and perhaps most crucial benefit of PP in a model-centered approach to assured autonomy is that prediction error provides a continuous assessment of the accuracy of the model and a systematic purchase on situations that warrant further investigation. Small prediction errors indicate that all is proceeding satisfactorily (we will see later that this may not be so and we need to separately verify the absence of systemic misinterpretations, but the basic inference is correct). Large prediction errors constitute *surprise* and indicate a failure of some kind, a flaw in the model, or an unexpected development in the world. Prediction error thus provides a single, uniform method for runtime verification and failure detection, so that assurance for the autonomous system is itself autonomous. For example, disappearance of previously detected objects because they are occluded by an overturned truck will trigger surprise even if our object detector fails to recognize the truck (cf. Tesla crash, Taiwan 1 Jun 20).

There are just three possible responses to surprise: we can adjust the sensor(s), adjust the model, or adjust the world. Selection and coordination among these is best performed by a separate function that has access to knowledge and more comprehensive methods for reasoning about the world, as we now describe.

2.2 Dual-Process Architecture

Suppose we are driving on a freeway when the object detector of a camera indicates a bicycle in front of us. In fact, what has been seen is a picture of a bicycle painted on the rear of a van. The object detector has seen the van in earlier frames and correctly labeled it but, now that we are closer, it is able to resolve the picture and labels the object as a bicycle, or perhaps as a van with confidence x and bicycle with confidence y. PP will have predicted it is definitely a van, so we have a large prediction error and need to decide how to proceed.

Now, we are on a freeway and the local "Laws and Rules of the Road" specify that bicycles are not allowed on freeways (this is not always so, but we will ignore special cases for the purpose of illustration). If our object detector had access to this information it could make the inference that the detected object cannot be a bicycle and must be a van. Such knowledge and methods of inference seem a useful augmentation to model-centered assurance—the more the system knows, the less it needs to sense—but pose the question how they should be incorporated into the architecture. The capabilities seem too general and universal to be added to the interpretation functions of individual sensors, so it seems they could be added to the model construction function. However, we now think of model construction as a predictive processing loop that sends predictions to sensors and updates the model according to prediction errors. The higher level knowledge and inferences that we now contemplate seem asynchronous to this loop and more focused on analysis and augmentation of the world model and the response to surprise. Accordingly, we propose that these capabilities—essentially, the deductive AI functions of the system—are located in a separate "model refinement" function that is shown as dark pink in Fig. 1. Furthermore, we suggest that it is useful to think of the two methods of model update as occurring on two levels: basic (PP) model construction at Level 1, and more inferential refinement at the higher Level 2.

Just as predictive processing has independent engineering justification, but also happens to coincide with a current theory of human cognition, so the two-level architecture coincides with a related "dual-process" theory [27,28], recently popularized by Kahneman as separate "fast and slow" systems of human thought [29]. In human cognition, *System 1* is unconscious, fast, and specialized for routine tasks; *System 2* is conscious, slow, easily fatigued, and capable of deliberation and reasoning—it is what we mean by "thinking." These correspond to Levels 1 and 2 in the architecture of Fig. 1.

We have implicitly suggested that Level 1 Model Construction maintains a single assured world model (that may have several components) and that Level 2 Model Refinement embellishes this. However, it is believed that humans maintain a hierarchy of models [24,26] and this seems a good approach for autonomous systems as well. The idea is that a PP loop operates between each adjacent pair (in the hierarchy) of models, so that the lower level is like a sensor for the upper level, with priority and update frequency determined by the size of prediction errors. We will not pursue this more elaborate architecture here, but it seems an interesting prospect for future consideration.

In general, the purpose of Level 2 Model Refinement is to respond to and minimize "surprise" and one way to approach the latter is to augment the world model with additional information that encodes "situation awareness." This can be the result of hypothetical and counterfactual reasoning, so that things not seen but "that might be there" are explicitly added to the model as "ghosts" (objects with low probability). For example, by reasoning about lines of sight and occluded vision, we could deduce that "we will be unable to see any vehicle that might be behind that truck." We could then add a ghost vehicle to the occupancy grid location behind the truck and this will influence the action functions. When the truck no longer occludes our vision, sensors will confirm or deny presence of the ghost, but neither outcome will constitute a surprise or a hazard because the action functions will not have committed to a choice that would be dangerous for either outcome. Dually, we might determine that a detected vehicle may be unable to see us because a truck is in the way, and this can be incorporated in the model as increased uncertainty on the likely motions of the vehicle concerned.

In another example, detection of many brake lights up ahead can be used to infer some kind of problem and this will be represented as increased uncertainty in the world model. In this way, what might otherwise be the surprising manifestation of an unanticipated situation will instead develop as gradual changes in uncertainty or the resolution of ghosts into real objects. Observe that these refinements to the world model make it more conservative: that is, they reduce the options available to action functions, so that faults in this type of Level 2 reasoning cannot make the world model less safe.

Level 2 aims to minimize surprise but, when surprise happens, it must deal with it. We noted earlier that there are three possible responses: we can adjust the sensors, the model, or the world (or a combination of these). When surprise takes the form of a large prediction error from one sensor, or group of similar sensors, it is reasonable to assume it is a temporary glich in sensing (e.g., reflections confusing radar) or interpretation (e.g., adversarial-type inputs) and to ignore the afflicted sensor(s) for a while and trust to local continuity in the world, or diversity and redundancy among other sensors, to maintain an accurate model.

Alternatively, analysis may suggest that prediction errors are due to a persistent cause in the external world, such as cameras dazzled by the sun. A suitable response might then be to change lanes to get in the shade of a nearby truck—in other words, to make a change in the world (specifically, our location in it).

If prediction errors persist, or afflict an entire class of sensor, then it may be a hardware or system problem—in which case Level 2 may coordinate with the fault tolerance functions of the underlying platform—or it may be some environmental challenge, such as fog. In this situation, lane-detection by cameras may fail, but Level 2 may be able to substitute some alternative method of "sensing" for model update, such as inferring location of the traffic lanes from the motions of neighboring cars as revealed by radar and proximity sensors.

When many sensors register large prediction errors, then the cause of the surprise is likely to be that the world did not evolve as the model predicted, and the appropriate response is to make changes in the model so that it conforms to information from the sensors. In extreme cases, this may require rebuilding the model from scratch, using some frames of bottom-up anti-causal interpretation.

We have motivated and introduced the two key aspects of our model-centered approach for autonomous systems: predictive processing and dual-process architecture. We now develop an outline assurance argument.

3 Toward an Assurance Argument

In routine operation, an autonomous system employing our model-centered approach will maintain a comprehensive naturalistic model of the world, possibly comprised of several components including (e.g., for cars) a detected objects list and an occupancy grid. The model is updated in an automated and regular manner using predictive processing, which provides continuous feedback on the quality of the model and available sensor data. For each sensor, the system will generate a prediction for the interpretation of its next frame. This can allow the sensor to focus its resources and interpretation in the most productive manner. Each sensor will report its prediction error; a small error indicates the model is aligned with the world and will cause a modest update to the model; a large error indicates a "surprise" that could be due to either a fault in the sensor or its interpretation, a flaw in the model, or an unexpected development in the world.

The most basic branch in an assurance case for this architecture is to be sure that small prediction errors really do guarantee that the model is a safely approximate representation of the world. The primary hazard to this claim is that the system's perception of the world could be systematically distorted—a hallucination. For example it could be blind to red cars: sensors will report no red cars, leading to construction of a world model without red cars; this will generate predictions lacking red cars that are confirmed by the sensors. So all seems well—until we collide with a red car, or are otherwise surprised by their existence. We refer to these perception faults as *systemic* and their characteristic is that they occur in some significant region of the input space and for significant periods of time, leading to misperception of the world. Other systemic faults might concern measurement rather than existence: for example, a sensor might displace objects 10 ft to the left or miscalculate bounding boxes by 10%; and others might concern identification, such as misreading stop signs as something else.

Systemic faults must be excluded or eliminated during development. A reasonable way to detect residual systemic faults—and to provide evidence for their absence—is by testing, and a plausible way to perform this is in simulation. We generate a simulated model of the world and a rendering of it is presented to the sensor interpretation and model construction functions under test; the model they construct is then compared to the original. A few million miles of simulated travel through representative scenarios [1] with some injection of degradation (e.g., rain and fog) into the rendering, could provide reasonable assurance for the absence of systemic faults. Note that this is not the same as "collecting miles" as we are looking only for systemic faults, whereas those who collect miles are also seeking exceptionally rare "corner case" and "fat tail" failures.

Assuming adequate evidence that small prediction errors guarantee the model is safely approximate, the next major branch in the assurance case is to show that the responses to large prediction errors (i.e., surprise) maintain the property. This divides into case analysis over the causes of surprise. First is a hardware fault of some kind, and we assume that this is safely handled by the underlying fault-tolerant platform (possibly in coordination with Level 2 model refinement).

Next is some fault in sensor interpretation or model construction that is localized in space and time (i.e., not systemic); we call these *local* perception faults and rely on assumed local continuity of the world to justify freezing or extrapolating afflicted values for a few frames; alternatively, if redundant sensors are available, we assume failure independence and favor these. Failure independence seems a plausible assumption, especially for sensors using different phenomena.

When prediction errors persist for more than a few frames, or are shared by multiple sensors, then we have a genuine surprise (the model is inaccurate, or the world has not evolved as expected) and Level 2 functions will orchestrate repair or regeneration of the model, or take action in the world, as described in the previous section. We will not drive this sketch of an assurance case into these details as they will be specific to each system. However, we note that the primary purpose of the AI functions at Level 2 is to resolve uncertainties in sensor interpretation (e.g., the bicycle/van example mentioned earlier), and to reduce future surprise through augmentations to the model that anticipate its evolution and promote situation awareness. These activities all increase the conservatism of the model and can only enhance the safety of its approximation to the real world.

This safety argument is quite different than could be constructed for traditional bottom-up sensor interpretation and fusion: predictive processing provides an intrinsic structure to the case analysis that suggests the argument can be made deductive [30]. Of course, much remains to be investigated, including a search for defeaters and quantifiable assessment of confidence and risks. A critical next step will be to assess confidence that surprises are always generated when they should be, which might be done by fault injection (e.g., perturbing interpretation of sensors, or the modeled world presented to them) in simulation.

4 Conclusion

We have described an architecture for autonomous systems that promotes their safety and assurance. The focus of the architecture is the "safely approximate" accuracy of an assured world model maintained by the perception functions. Given this model, the action functions can be guarded by monitors driven by traditional software and assured in traditional ways.

In the proposed architecture, sensor interpretation is driven top-down from the model—the reverse of the usual direction—using predictive processing. Small prediction errors confirm accuracy of the model and sensor interpretation, given that testing confirms absence of systemic faults. Large prediction errors indicate either a fault in sensor interpretation, or a departure between the model and the world (a surprise). The uniformity of prediction error as the single basis for runtime assurance and fault detection means that assurance is itself autonomous. AI functions located at a second level provide situation awareness and other measures to reduce future surprise and to deal with it when it occurs.

Although this dual-process architecture with predictive processing is derived by engineering considerations, it corresponds with current theories of human cognition and suggests fruitful comparisons. Mathematical formulations of predictive processing are well-developed in cognitive science [26] and in machine learning, where it is associated with "generative" and "variational" methods [16].

In current work, we are performing experimental evaluations of the approach and plan to report our experience and propose key challenges.

Acknowledgments. We thank the reviewers for their constructive comments, and Bev Littlewood of City, University of London, and Wilfried Steiner of TTTech Vienna for their challenges and discussion. The work was funded by DARPA contract FA8750-19-C-0089. We would also like to acknowledge support from the US Army Research Laboratory Cooperative Research Agreement W911NF-17-2-0196, and National Science Foundation (NSF) grant 1740079. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

1. Thorn, E., Kimmel, S., Chaka, M.: A framework for automated driving system testable cases and scenarios. DOT HS 812 623, NHTSA, Washington DC (2018)
2. Kalra, N., Paddock, S.M.: Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Transp. Res. Part A Policy Pract. **94**, 182–193 (2016)
3. ASTM: Standard Practice for Methods to Safely Bound Flight Behavior of Unmanned Aircraft Systems Containing Complex Functions (2017). ASTM F3269–17
4. Shalev-Shwartz, S., Shammah, S., Shashua, A.: On a formal model of safe and scalable self-driving cars. arXiv preprint arXiv:1708.06374 (2017)
5. NHTSA: Automated driving systems 2.0: A vision for safety. DOT HS 812 442, Washington DC (2018)
6. Lin, S.C., et al.: The architectural implications of autonomous driving: constraints and acceleration. ACM SIGPLAN Notices **53**, 751–766 (2018)
7. Koller, D., Daniilidis, K., Thórhallson, T., Nagel, H.-H.: Model-based object tracking in traffic scenes. In: Sandini, G. (ed.) ECCV 1992. LNCS, vol. 588, pp. 437–452. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55426-2_49
8. Pearl, J., Mackenzie, D.: The Book of Why: The New Science of Cause and Effect. Basic Books, New York (2018)
9. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv:1312.6199 (2013)
10. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1765–1773 (2017)
11. Brown, T.B., et al.: Adversarial patch. arXiv preprint arXiv:1712.09665 (2017)
12. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: bypassing ten detection methods. In: Proceedings of the 10th ACM W'shop on AI & Security, pp. 3–14 (2017)
13. Shamir, A., Safran, I., Ronen, E., Dunkelman, O.: A simple explanation for the existence of adversarial examples with small Hamming distance. arXiv preprint arXiv:1901.10861 (2019)
14. Kilbertus, N., Parascandolo, G., Schölkopf, B.: Generalization in anti-causal learning. arXiv preprint arXiv:1812.00524 (2018)
15. Pimentel, M.A.F., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Sig. Process. **99**, 215–249 (2014)
16. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114 (2013)
17. Jha, S., Jang, U., Jha, S., Jalaian, B.: Detecting adversarial examples using data manifolds. In: MILCOM 2018, pp. 547–552. IEEE (2018)
18. Ju, C., Bibaut, A., van der Laan, M.: The relative performance of ensemble methods with deep convolutional neural networks for image classification. J. Appl. Stat. **45**, 2800–2818 (2018)
19. National Transportation Safety Board: Vehicle Automation Report; Tempe, AZ (2019). HWY18MH010
20. Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. Commun. ACM **15**, 11–15 (1972)
21. Spratling, M.W.: A review of predictive coding algorithms. Brain Cogn. **112**, 92–97 (2017)

22. von Helmholtz, H.: Handbuch der Physiologischen Optik III, vol. 9. Verlag von Leopold Voss, Leipzig, Germany (1867)
23. Wiese, W., Metzinger, T.K.: Vanilla PP for philosophers: a primer on Predictive Processing. MIND Group, Frankfurt am Main (2017). Many papers: https://predictive-mind.net/papers
24. Clark, A.: Whatever next? Predictive brains, situated agents, and the future of cognitive science. Behav. Brain Sci. **36**, 181–204 (2013)
25. Hohwy, J.: The Predictive Mind. Oxford University Press, Oxford (2013)
26. Friston, K.: The free-energy principle: A unified brain theory? Nat. Rev. Neurosci. **11**, 127 (2010)
27. Frankish, K.: Dual-process and dual-system theories of reasoning. Philos. Compass **5**, 914–926 (2010)
28. Evans, J.S.B.T., Stanovich, K.E.: Dual-process theories of higher cognition: Advancing the debate. Perspect. Psychol. Sci. **8**, 223–241 (2013)
29. Kahneman, D.: Thinking. Fast and Slow. Farrar, Straus and Giroux (2011)
30. Bloomfield, R., Rushby, J.: Assurance 2.0: A Manifesto. arXiv:2004.10474 (2020)